Network Working Group                                         M. Daniele
Request for Comments: 2257                 Digital Equipment Corporation
Category: Standards Track                                      B. Wijnen
                                   T.J. Watson Research Center, IBM Corp.
                                                      D. Francisco, Ed.
                                                    Cisco Systems, Inc.
                                                           January 1998

# Agent Extensibility (AgentX) Protocol Version 1

## Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

## Copyright Notice

Copyright (C) The Internet Society (1998). All Rights Reserved.

## Table of Contents

# 1. Introduction

This memo defines a standardized framework for extensible SNMP agents. It defines processing entities called master agents and subagents, a protocol (AgentX) used to communicate between them, and the elements of procedure by which the extensible agent processes SNMP protocol messages.

# 2. The SNMP Framework

A management system contains: several (potentially many) nodes, each with a processing entity, termed an agent, which has access to management instrumentation; at least one management station; and, a management protocol, used to convey management information between the agents and management stations. Operations of the protocol are carried out under an administrative framework which defines authentication, authorization, access control, and privacy policies.

Management stations execute management applications which monitor and control managed elements. Managed elements are devices such as hosts, routers, terminal servers, etc., which are monitored and controlled via access to their management information.

Management information is viewed as a collection of managed objects, residing in a virtual information store, termed the Management Information Base (MIB). Collections of related objects are defined in MIB modules. These modules are written using a subset of OSI's Abstract Syntax Notation One (ASN.1) [1], termed the Structure of Management Information (SMI) (see RFC 1902 [2]).

## 2.1. A Note on Terminology

The term "variable" refers to an instance of a non-aggregate object type defined according to the conventions set forth in the SMI (RFC 1902, [2]) or the textual conventions based on the SMI (RFC 1903 [3]). The term "variable binding" normally refers to the pairing of the name of a variable and its associated value. However, if certain kinds of exceptional conditions occur during processing of a retrieval request, a variable binding will pair a name and an indication of that exception.

A variable-binding list is a simple list of variable bindings.

The name of a variable is an OBJECT IDENTIFIER, which is the concatenation of the OBJECT IDENTIFIER of the corresponding object type together with an OBJECT IDENTIFIER fragment identifying the instance. The OBJECT IDENTIFIER of the corresponding object-type is called the OBJECT IDENTIFIER prefix of the variable. For the purpose of exposition, the original Internet-standard

Network Management Framework, as described in RFCs 1155 (STD 16), 1157 (STD 15), and 1212 (STD 16), is termed the SNMP version 1 framework (SNMPv1). The current framework, as described in RFCs 1902-1908, is termed the SNMP version 2 framework (SNMPv2).

## 3. Extending the MIB

New MIB modules that extend the Internet-standard MIB are continuously being defined by various IETF working groups. It is also common for enterprises or individuals to create or extend enterprise-specific or experimental MIBs.

As a result, managed devices are frequently complex collections of manageable components that have been independently installed on a managed node. Each component provides instrumentation for the managed objects defined in the MIB module(s) it implements.

Neither the SNMP version 1 nor version 2 framework describes how the set of managed objects supported by a particular agent may be changed dynamically.

### 3.1. Motivation for AgentX

This very real need to dynamically extend the management objects within a node has given rise to a variety of "extensible agents", which typically comprise

- a "master" agent that is available on the standard transport address and that accepts SNMP protocol messages

- a set of "subagents" that each contain management instrumentation

- a protocol that operates between the master agent and subagents, permitting subagents to "connect" to the master agent, and the master agent to multiplex received SNMP protocol messages amongst the subagents.

- a set of tools to aid subagent development, and a runtime (API) environment that hides much of the protocol operation between a subagent and the master agent.

The wide deployment of extensible SNMP agents, coupled with the lack of Internet standards in this area, makes it difficult to field SNMP-manageable applications. A vendor may have to support several different subagent environments (APIs) in order to support different target platforms.

It can also become quite cumbersome to configure subagents and (possibly multiple) master agents on a particular managed node.

Specifying a standard protocol for agent extensibility (AgentX) provides the technical foundation required to solve both of these problems. Independently developed AgentX-capable master agents and subagents will be able to interoperate at the protocol level. Vendors can continue to differentiate their products in all other respects.

## 4. AgentX Framework

Within the SNMP framework, a managed node contains a processing entity, called an agent, which has access to management information.

Within the AgentX framework, an agent is further defined to consist of

- a single processing entity called the master agent, which sends and receives SNMP protocol messages in an agent role (as specified by the SNMP version 1 and version 2 framework documents) but typically has little or no direct access to management information.

- 0 or more processing entities called subagents, which are "shielded" from the SNMP protocol messages processed by the master agent, but which have access to management information.

The master and subagent entities communicate via AgentX protocol messages, as specified in this memo. Other interfaces (if any) on these entities, and their associated protocols, are outside the scope of this document. While some of the AgentX protocol messages appear similar in syntax and semantics to the SNMP, bear in mind that AgentX is not SNMP.

The internal operations of AgentX are invisible to an SNMP entity operating in a manager role. From a manager's point of view, an extensible agent behaves exactly as would a non-extensible (monolithic) agent that has access to the same management instrumentation.

This transparency to managers is a fundamental requirement of AgentX, and is what differentiates AgentX subagents from SNMP proxy agents.

## 4.1. AgentX Roles

An entity acting in a master agent role performs the following functions:

- Accepts AgentX session establishment requests from subagents.

- Accepts registration of MIB regions by subagents.

- Sends and accepts SNMP protocol messages on the agent's specified transport addresses.

- Implements the agent role Elements of Procedure specified for the administrative framework applicable to the SNMP protocol message, except where they specify performing management operations. (The application of MIB views, and the access control policy for the managed node, are implemented by the master agent.)

- Provides instrumentation for the MIB objects defined in RFC 1907 [5], and for any MIB objects relevant to any administrative framework it supports.

- Sends and receives AgentX protocol messages to access management information, based on the current registry of MIB regions.

- Forwards notifications on behalf of subagents.

An entity acting in a subagent role performs the following functions:

- Initiates an AgentX session with the master agent.

- Registers MIB regions with the master agent.

- Instantiates managed objects.

- Binds OIDs within its registered MIB regions to actual variables.

- Performs management operations on variables.

- Initiates notifications.

## 4.2 Applicability

It is intended that this memo specify the smallest amount of required behavior necessary to achieve the largest benefit, that is, to cover a very large number of possible MIB implementations and configurations with minimum complexity and low "cost of entry".

This section discusses several typical usage scenarios.

1) Subagents implement separate MIB modules--for example, subagent A implements "mib-2", subagent b implements "host-resources".

   It is anticipated that this will be the most common subagent configuration.

2) Subagents implement rows in a "simple table". A simple table is one in which row creation is not specified, and for which the MIB does not define an object that counts entries in the table. Examples of simple tables are rdbmsDbTable, udpTable, and hrSWRunTable.

   This is the most commonly defined type of MIB table, and probably represents the next most typical configuration that AgentX would support.

3) Subagents share MIBs along non-row partitions. Subagents register "chunks" of the MIB that represent multiple rows, due to the nature of the MIB's index structure. Examples include registering ipNetToMediaEntry.n, where n represents the ifIndex value for an interface implemented by the subagent, and tcpConnEntry.a.b.c.d, where a.b.c.d represents an IP address on an interface implemented by the subagent.

AgentX supports these three common configurations, and all permutations of them, completely. The consensus is that they comprise a very large majority of current and likely future uses of multi-vendor extensible agent configurations.

4) Subagents implement rows in "complex tables". Complex tables here are defined as tables permitting row creation, or whose MIB also defines an object that counts entries in the table. Examples include the MIB-2 ifTable (due to ifNumber), and the RMON historyControlTable.

The subagent that implements such a counter object (like ifNumber) must go beyond AgentX to correctly implement it. This is an implementation issue (and most new MIB designs no longer include such objects).

To implement row creation in such tables, at least one AgentX subagent must register at a point "higher" in the OID tree than an individual row (per AgentX's dispatching procedure). Again, this is an implementation issue.

Scenarios in this category were thought to occur somewhat rarely in configurations where subagents are independently implemented by different vendors. The focus of a standard protocol, however, must be in just those areas where multi- vendor interoperability must be assured.

Note that it would be inefficient (due to AgentX registration overhead) to share a table among AgentX subagents if the table contains very dynamic instances, and each subagent registers fully qualified instances. ipRouteTable could be an example of such a table in some environments.

## 4.3. Design Features of AgentX

The primary features of the design described in this memo are:

1) A general architectural division of labor between master agent and subagent: The master agent is MIB ignorant and SNMP omniscient, while the subagent is SNMP ignorant and MIB omniscient (for the MIB variables it instantiates). That is, master agents, exclusively, are concerned with SNMP protocol operations and the translations to and from AgentX protocol operations needed to carry them out; subagents are exclusively concerned with management instrumentation; and neither should intrude on the other's territory.

2) A standard protocol and "rules of engagement" to enable interoperability between management instrumentation and extensible agents.

3) Mechanisms for independently developed subagents to integrate into the extensible agent on a particular managed node in such a way that they need not be aware of any other existing subagents.

4) A simple, deterministic registry and dispatching algorithm. For a given extensible agent configuration, there is a single subagent who is "authoritative" for any particular region of the MIB (where "region" may extend from an entire MIB down to a single object-instance).

5) Performance considerations. It is likely that the master agent and all subagents will reside on the same host, and in such cases AgentX is more a form of inter-process communication than a traditional communications protocol.

   Some of the design decisions made with this in mind include:

   - 32-bit alignment of data within PDUs

   - Native byte-order encoding by subagents

   - Large AgentX PDU payload sizes.

## 4.4 Non-Goals

1) Subagent-to-subagent communication. This is out of scope, due to the security ramifications and complexity involved.

2) Subagent access (via the master agent) to MIB variables. This is not addressed, since various other mechanisms are available and it was not a fundamental requirement.

3) The ability to accommodate every conceivable extensible agent configuration option. This was the most contentious aspect in the development of this protocol. In essence, certain features currently available in some commercial extensible agent products are not included in AgentX. Although useful or even vital in some implementation strategies, the rough consensus was that these features were not appropriate for an Internet Standard, or not typically required for independently developed subagents to coexist. The set of supported extensible agent configurations is described above, in Section 4.2.

Some possible future version of the AgentX protocol may provide coverage for one or more of these "non-goals" or for new goals that might be identified after greater deployment experience.

# 5. AgentX Encodings

AgentX PDUs consist of a common header, followed by PDU-specific data of variable length. Unlike SNMP PDUs, AgentX PDUs are not encoded using the BER (as specified in ISO 8824 [1]), but are transmitted as a contiguous byte stream. The data within this stream is organized to provide natural alignment with respect to the start of the PDU, permitting direct (integer) access by the processing entities.

The first four fields in the header are single-byte values. A bit (NETWORK_BYTE_ORDER) in the third field (h.flags) is used to indicate the byte ordering of all multi-byte integer values in the PDU, including those which follow in the header itself. This is described in more detail in Section 6.1, "AgentX PDU Header", below.

PDUs are depicted in this memo using the following convention (where byte 1 is the first transmitted byte):

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  byte 1       |  byte 2       |  byte 3       |  byte 4       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  byte 5       |  byte 6       |  byte 7       |  byte 8       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields marked "<reserved>" are reserved for future use and must be zero-filled.

## 5.1. Object Identifier

An object identifier is encoded as a 4-byte header, followed by a variable number of contiguous 4-byte fields representing sub-identifiers. This representation (termed Object Identifier) is as follows:

Object Identifier

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  n_subid      |  prefix       |  include      |  <reserved>   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       sub-identifier #1                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       sub-identifier #n_subid                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Object Identifier header fields:

n_subid

The number (0-128) of sub-identifiers in the object identifier. An ordered list of "n_subid" 4-byte sub-identifiers follows the 4-byte header.

prefix

An unsigned value used to reduce the length of object identifier encodings. A non-zero value "x" is interpreted as the first sub-identifier after "internet" (1.3.6.1), and indicates an implicit prefix "internet.x" to the actual sub-identifiers encoded in the Object Identifier. For example, a prefix field value 2 indicates an implicit prefix "1.3.6.1.2". A value of 0 in the prefix field indicates there is no prefix to the sub-identifiers.

include

Used only when the Object Identifier is the start of a SearchRange, as described in section 5.2.

A null Object Identifier consists of the 4-byte header with all bytes set to 0.

Examples:

sysDescr.0 (1.3.6.1.2.1.1.1.0)

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 4             | 2             | 0             | 0             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 1                                                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 1                                                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 1                                                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0                                                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

1.2.3.4

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 4             | 0             | 0             | 0             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 1                                                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 2                                                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 3                                                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 4                                                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

## 5.2. SearchRange

A SearchRange consists of two Object Identifiers. In its communication with a subagent, the master agent uses a SearchRange to identify a requested variable binding, and, in GetNext and GetBulk operations, to set an upper bound on the names of managed object instances the subagent may send in reply.

The first Object Identifier in a SearchRange (called the starting OID) indicates the beginning of the range. It is frequently (but not necessarily) the name of a requested variable binding.

The "include" field in this OID's header is a boolean value (0 or 1) indicating whether or not the starting OID is included in the range.

The second object identifier indicates the non-inclusive end of the range, and its "include" field is always 0.

Example: To indicate a search range from 1.3.6.1.2.1.25.2 (inclusive) to 1.3.6.1.2.1.25.2.1 (exclusive), the SearchRange would be

```
(start)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 3             | 2             | 1             |       0       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 1                                                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 25                                                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 2                                                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

(end)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 4             | 2             | 0             |       0       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 1                                                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 25                                                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 2                                                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 1                                                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

A SearchRangeList is a contiguous list of SearchRanges.

## 5.3. Octet String

An octet string is represented by a contiguous series of bytes, beginning with a 4-byte integer whose value is the number of octets in the octet string, followed by the octets themselves. This representation is termed an Octet String. If the last octet does not end on a 4-byte offset from the start of the Octet String, padding bytes are appended to achieve alignment of following data. This padding must be added even if the Octet String is the last item in the PDU. Padding bytes must be zero filled.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Octet String Length (L)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Octet 1      |  Octet 2      |   Octet 3     |   Octet 4     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Octet L - 1  |  Octet L      |       Padding (as required)   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

A null Octet String consists of a 4-byte length field set to 0.

## 5.4. Value Representation

Variable bindings may be encoded within the variable-length portion of some PDUs. The representation of a variable binding (termed a VarBind) consists of a 2-byte type field, a name (Object Identifier), and the actual value data.

VarBind

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          v.type               |          <reserved>           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

(v.name)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  n_subid      |  prefix       |      0        |       0       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       sub-identifier #1                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       sub-identifier #n_subid                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
   (v.data)
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       data                                    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       data                                    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

VarBind fields:

v.type

Indicates the variable binding's syntax, and must be one of the following values:

```
                  Integer                  (2),
                  Octet String             (4),
                  Null                     (5),
                  Object Identifier        (6),
                  IpAddress               (64),
                  Counter32               (65),
                  Gauge32                 (66),
                  TimeTicks               (67),
                  Opaque                  (68),
                  Counter64               (70),
                  noSuchObject           (128),
                  noSuchInstance         (129),
                  endOfMibView           (130)
```

v.name

The Object Identifier which names the variable.

v.data

The actual value, encoded as follows:

- Integer, Counter32, Gauge32, and TimeTicks are encoded as 4 contiguous bytes.  If the NETWORK_BYTE_ORDER bit is set in h.flags, the bytes are ordered most significant to least significant, otherwise they are ordered least significant to most significant.

- Counter64 is encoded as 8 contiguous bytes.  If the NETWORK_BYTE_ORDER bit is set in h.flags, the bytes are ordered most significant to least significant, otherwise they are ordered least significant to most significant.

- Object Identifiers are encoded as described in section 5.1, Object Identifier.

- IpAddress, Opaque, and Octet String are all octet strings and are encoded as described in section 5.3, Octet String.

  Value data always follows v.name whenever v.type is one of the above types. These data bytes are present even if they will not be used (as, for example, in certain types of index allocation).

- Null, noSuchObject, noSuchInstance, and endOfMibView do not contain any encoded value. Value data never follows v.name in these cases.

Note that the VarBind itself does not contain the value size. That information is implied for the fixed-length types, and explicitly contained in the encodings of variable-length types (Object Identifier and Octet String).

A VarBindList is a contiguous list of VarBinds. Within a VarBindList, a particular VarBind is identified by an index value. The first VarBind in a VarBindList has index value 1, the second has index value 2, and so on.

# 6. Protocol Definitions

## 6.1. AgentX PDU Header

The AgentX PDU header is a fixed-format, 20-octet structure:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   h.version   |    h.type     |    h.flags    |  <reserved>   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          h.sessionID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        h.transactionID                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          h.packetID                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        h.payload_length                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

An AgentX PDU header contains the following fields:

h.version

The version of the AgentX protocol (1 for this memo).

h.type

The PDU type; one of the following values:

```
agentx-Open-PDU              (1),
agentx-Close-PDU             (2),
agentx-Register-PDU          (3),
agentx-Unregister-PDU        (4),
agentx-Get-PDU               (5),
agentx-GetNext-PDU           (6),
agentx-GetBulk-PDU           (7),
agentx-TestSet-PDU           (8),
agentx-CommitSet-PDU         (9),
agentx-UndoSet-PDU          (10),
agentx-CleanupSet-PDU       (11),
agentx-Notify-PDU           (12),
agentx-Ping-PDU             (13),
agentx-IndexAllocate-PDU    (14),
agentx-IndexDeallocate-PDU  (15),
agentx-AddAgentCaps-PDU     (16),
agentx-RemoveAgentCaps-PDU  (17),
agentx-Response-PDU         (18)
```

h.flags

A bitmask, with bit 0 the least significant bit. The bit definitions are as follows:

| Bit | Definition |
|---|---|
| 0 | INSTANCE_REGISTRATION |
| 1 | NEW_INDEX |
| 2 | ANY_INDEX |
| 3 | NON_DEFAULT_CONTEXT |
| 4 | NETWORK_BYTE_ORDER |
| 5-7 | (reserved) |

The NETWORK_BYTE_ORDER bit applies to all multi-byte integer values in the entire AgentX packet, including the remaining header fields. If set, then network byte order (most significant byte first; "big endian") is used. If not set, then least significant byte first ("little endian") is used.

The NETWORK_BYTE_ORDER bit applies to all AgentX PDUs.

The NON_DEFAULT_CONTEXT bit is used only in the AgentX PDUs described in section 6.1.1.

The NEW_INDEX and ANY_INDEX bits are used only within the agentx-IndexAllocate-, and -IndexDeallocate-PDUs.

The INSTANCE_REGISTRATION bit is used only within the agentx-Register-PDU.

h.sessionID

The session ID uniquely identifies a session over which AgentX PDUs are exchanged between a subagent and the master agent. The session ID has no significance and no defined value in the agentx-Open-PDU sent by a subagent to open a session with the master agent; in this case, the master agent will assign a unique sessionID that it will pass back in the corresponding agentx-Response-PDU. From that point on, that same sessionID will appear in every AgentX PDU exchanged over that session between the master and the subagent. A subagent may establish multiple AgentX sessions by sending multiple agentx-Open-PDUs to the master agent.

In master agents that support multiple transport protocols, the sessionID should be globally unique rather than unique just to a particular transport.

h.transactionID

The transaction ID uniquely identifies, for a given session, the single SNMP management request (and single SNMP PDU) with which an AgentX PDU is associated. If a single SNMP management request results in multiple AgentX PDUs being sent by the master agent with the same sessionID, each of these AgentX PDUs must contain the same transaction ID; conversely, AgentX PDUs sent during a particular session, that result from distinct SNMP management requests, must have distinct transaction IDs within the limits of the 32-bit field).

Note that the transaction ID is not the same as the SNMP PDU's request-id (as described in section 4.1 of RFC 1905 [4]), nor can it be, since a master agent might receive SNMP requests with the same request-ids from different managers.

The transaction ID has no significance and no defined value in AgentX administrative PDUs, i.e., AgentX PDUs that are not associated with an SNMP management request.

h.packetID

A packet ID generated by the sender for all AgentX PDUs except the agentx-Response-PDU. In an agentx-Response-PDU, the packet ID must be the same as that in the received AgentX PDU to which it is a response. A master agent might use this field to associate subagent response PDUs with their corresponding request PDUs. A subagent might use this field to correlate responses to multiple (batched) registrations.

h.payload_length

The size in octets of the PDU contents, excluding the 20-byte header. As a result of the encoding schemes and PDU layouts, this value will always be either 0, or a multiple of 4.

### 6.1.1. Context

In the SNMPv1 or v2c frameworks, the community string may be used as an index into a local repository of configuration information that may include community profiles or more complex context information. Future versions of the SNMP will likely formalize this notion of "context".

AgentX provides a mechanism for transmitting a context specification within relevant PDUs, but does not place any constraints on the content of that specification.

An optional context field may be present in the agentx-Register-, UnRegister-, AddAgentCaps-, RemoveAgentCaps-, Get-, GetNext-, GetBulk-, IndexAllocate-, IndexDeallocate-, Notify-, TestSet-, and Ping- PDUs.

If the NON_DEFAULT_CONTEXT bit in the AgentX header field h.flags is clear, then there is no context field in the PDU, and the operation refers to the default context.

If the NON_DEFAULT_CONTEXT bit is set, then a context field immediately follows the AgentX header, and the operation refers to that specific context. The context is represented as an Octet String. There are no constraints on its length or contents.

Thus, all of these AgentX PDUs (that is, those listed immediately above) refer to, or "indicate" a context, which is either the default context, or a non-default context explicitly named in the PDU.

## 6.2. AgentX PDUs

### 6.2.1. The agentx-Open-PDU

An agentx-Open-PDU is generated by a subagent to request establishment of an AgentX session with the master agent.

```
(AgentX header)

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| h.version (1) |  h.type (1)   |    h.flags    |  <reserved>   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          h.sessionID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        h.transactionID                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           h.packetID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        h.payload_length                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  o.timeout    |                     <reserved>                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

(o.id)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  n_subid      |  prefix       |       0       |  <reserved>   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             subidentifier #1                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...                                                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             subidentifier #n_subid                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

(o.descr)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Octet String Length (L)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Octet 1      |  Octet 2      |   Octet 3     |   Octet 4     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Octet L - 1  |  Octet L      |       Padding (as required)   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

An agentx-Open-PDU contains the following fields:

o.timeout

The length of time, in seconds, that a master agent should allow to elapse after dispatching a message to a subagent before it regards the subagent as not responding. This is a subagent-wide default value that may be overridden by values associated with specific registered MIB regions. The default value of 0 indicates that no subagent-wide value is requested.

o.id

An Object Identifier that identifies the subagent. Subagents that do not support such an notion may send a null Object Identifier.

o.descr

An Octet String containing a DisplayString describing the subagent.

### 6.2.2. The agentx-Close-PDU

An agentx-Close-PDU issued by either a subagent or the master agent terminates an AgentX session.

```
(AgentX header)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| h.version (1) |  h.type (2)   |    h.flags    |  <reserved>   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          h.sessionID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        h.transactionID                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           h.packetID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        h.payload_length                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  c.reason     |                   <reserved>                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

An agentx-Close-PDU contains the following field:

c.reason

An enumerated value that gives the reason that the master agent or subagent closed the AgentX session. This field may take one of the following values:

reasonOther(1)
None of the following reasons

reasonParseError(2)
Too many AgentX parse errors from peer

reasonProtocolError(3)
Too many AgentX protocol errors from peer

reasonTimeouts(4)
Too many timeouts waiting for peer

reasonShutdown(5)
Sending entity is shutting down

reasonByManager(6)
Due to Set operation; this reason code can be used only by the master agent, in response to an SNMP management request.

### 6.2.3. The agentx-Register-PDU

An agentx-Register-PDU is generated by a subagent for each region of the MIB variable naming tree (within one or more contexts) that it wishes to support.

```
 (AgentX header)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| h.version (1) |  h.type (3)   |    h.flags    |  <reserved>   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          h.sessionID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        h.transactionID                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           h.packetID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        h.payload_length                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
    (r.context) (OPTIONAL)
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                     Octet String Length (L)                   |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Octet 1      |  Octet 2      |   Octet 3     |   Octet 4     |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Octet L - 1  |  Octet L      |       Padding (as required)   |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  r.timeout    |  r.priority   | r.range_subid |  <reserved>   |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

    (r.region)
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  n_subid      |  prefix       |      0        |  <reserved>   |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |             sub-identifier #1                                 |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |             sub-identifier #n_subid                           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

    (r.upper_bound)
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |             optional upper-bound sub-identifier               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

An agentx-Register-PDU contains the following fields:

r.context

An optional non-default context.

r.timeout

The length of time, in seconds, that a master agent should allow to elapse after dispatching a message to a subagent before it regards the subagent as not responding.  r.timeout applies only to messages that concern MIB objects within r.region.  It overrides both the subagent-wide value (if any) indicated when the AgentX session with the master agent was established, and the master agent's default timeout.  The default value for r.timeout is 0 (no override).

r.priority

A value between 1 and 255, used to achieve a desired configuration when different subagents register identical or overlapping regions. Subagents with no particular knowledge of priority should register with the default value of 255 (lowest priority).

In the master agent's dispatching algorithm, smaller values of r.priority take precedence over larger values, as described in section 7.1.5.1.

r.region

An Object Identifier that, in conjunction with r.range_subid, indicates a region of the MIB that a subagent wishes to support. It may be a fully-qualified instance name, a partial instance name, a MIB table, an entire MIB, or ranges of any of these.

The choice of what to register is implementation-specific; this memo does not specify permissible values. Standard practice however is for a subagent to register at the highest level of the naming tree that makes sense. Registration of fully-qualified instances is typically done only when a subagent can perform management operations only on particular rows of a conceptual table.

If r.region is in fact a fully qualified instance name, the INSTANCE_REGISTRATION bit in h.flags must be set, otherwise it must be cleared. The master agent may save this information to optimize subsequent operational dispatching.

r.range_subid

Permits specifying a range in place of one of r.region's sub-identifiers. If this value is 0, no range is specified. Otherwise the "r.range_subid"-th sub-identifier in r.region is a range lower bound, and the range upper bound sub-identifier (r.upper_bound) immediately follows r.region.

This permits registering a conceptual row with a single PDU. For example, the following PDU would register row 7 of the RFC 1573 ifTable (1.3.6.1.2.1.2.2.1.1-22.7):

```
   (AgentX header)
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | h.version (1) |  h.type (3)   |    h.flags    |  <reserved>   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          h.sessionID                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                        h.transactionID                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          h.packetID                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                        h.payload_length                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   r.timeout   |  r.priority   | 5             |  <reserved>   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

   (r.region)
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | 6             |  2            |  0            |  <reserved>   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | 1                                                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | 2                                                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | 2                                                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | 1                                                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | 1                                                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | 7                                                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

   (r.upper_bound)
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | 22                                                            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

### 6.2.4. The agentx-Unregister-PDU

The agentx-Unregister-PDU is sent by a subagent to remove a previously registered MIB region from the master agent's OID space.

```
(AgentX header)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| h.version (1) |  h.type (4)   |    h.flags    |  <reserved>   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          h.sessionID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        h.transactionID                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           h.packetID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        h.payload_length                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

(u.context) OPTIONAL
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Octet String Length (L)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Octet 1      |  Octet 2      |   Octet 3     |   Octet 4     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Octet L - 1  |  Octet L      |       Padding (as required)   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    <reserved> |  u.priority   | u.range_subid |  <reserved>   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

(u.region)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  n_subid      |  prefix       |      0        |  <reserved>   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              sub-identifier #1                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              sub-identifier #n_subid                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

(u.upper_bound)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              optional upper-bound sub-identifier              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

An agentx-Unregister-PDU contains the following fields:

u.context

An optional non-default context.

u.priority

The priority at which this region was originally registered.

u.region

Indicates a previously-registered region of the MIB that a subagent no longer wishes to support.

### 6.2.5. The agentx-Get-PDU

```
(AgentX header)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| h.version (1) |  h.type (5)   |    h.flags    |  <reserved>   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          h.sessionID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        h.transactionID                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          h.packetID                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        h.payload_length                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

(g.context) OPTIONAL
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Octet String Length (L)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Octet 1      |  Octet 2      |   Octet 3     |   Octet 4     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Octet L - 1  |  Octet L      |       Padding (as required)   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

(g.sr)

(start 1)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  n_subid      |  prefix       |  include      |  <reserved>   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             sub-identifier #1                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             sub-identifier #n_subid                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

(end 1)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0             | 0             | 0             |      0        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...
(start n)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  n_subid      |  prefix       |  include      |  <reserved>   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             sub-identifier #1                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             sub-identifier #n_subid                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

(end n)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0             | 0             | 0             |      0        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

An agentx-Get-PDU contains the following fields:

g.context

An optional non-default context.

g.sr

A SearchRangeList containing the requested variables for this subagent.

### 6.2.6. The agentx-GetNext-PDU

```
(AgentX header)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| h.version (1) |  h.type (6)   |    h.flags    |  <reserved>   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          h.sessionID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        h.transactionID                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          h.packetID                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        h.payload_length                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

(g.context) OPTIONAL
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Octet String Length (L)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Octet 1      |  Octet 2      |   Octet 3     |   Octet 4     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Octet L - 1  |  Octet L      |       Padding (as required)   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

(g.sr)


(start 1)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  n_subid      |  prefix       |  include      |  <reserved>   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             sub-identifier #1                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             sub-identifier #n_subid                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

(end 1)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  n_subid      |  prefix       |      0        |  <reserved>   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             sub-identifier #1                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             sub-identifier #n_subid                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...

(start n)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  n_subid      |  prefix       |  include      |  <reserved>   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             sub-identifier #1                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             sub-identifier #n_subid                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

(end n)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  n_subid      |  prefix       |      0        |  <reserved>   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             sub-identifier #1                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             sub-identifier #n_subid                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...
```

### 6.2.7. The agentx-GetBulk-PDU

```
(AgentX header)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| h.version (1) |  h.type (7)   |    h.flags    |  <reserved>   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          h.sessionID                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        h.transactionID                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          h.packetID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        h.payload_length                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
   (g.context) OPTIONAL
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      Octet String Length (L)                  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Octet 1      |  Octet 2      |   Octet 3     |   Octet 4     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Octet L - 1  |  Octet L      |       Padding (as required)   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |             g.non_repeaters   |     g.max_repetitions         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

   (g.sr) as in agentx-GetNext-PDU above
   ...
```

6.2.8.  The agentx-TestSet-PDU

```
   (AgentX header)
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | h.version (1) |  h.type (8)   |    h.flags    |  <reserved>   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           h.sessionID                         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                         h.transactionID                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                            h.packetID                         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                         h.payload_length                      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

   (t.context) OPTIONAL
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      Octet String Length (L)                  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Octet 1      |  Octet 2      |   Octet 3     |   Octet 4     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Octet L - 1  |  Octet L      |       Padding (as required)   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

   (t.vb)
```

```
    (VarBind 1)
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |          v.type               |        <reserved>             |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  n_subid      |  prefix       |      0        |  <reserved>   |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                       sub-identifier #1                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                       sub-identifier #n_subid                 |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                       data                                    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                       data                                    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    ...

    (VarBind n)
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |          v.type               |        <reserved>             |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  n_subid      |  prefix       |      0        |  <reserved>   |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                       sub-identifier #1                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                       sub-identifier #n_subid                 |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                       data                                    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                       data                                    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

An agentx-TestSet-PDU contains the following fields:

t.context

An optional non-default context.

t.vb

A VarBindList containing the requested VarBinds for this subagent.

### 6.2.9. The agentx-CommitSet, -UndoSet, -CleanupSet PDUs

These PDUs consist of the AgentX header only.

The agentx-CommitSet-, -UndoSet-, and -Cleanup-PDUs are used in processing an SNMP SetRequest operation.

### 6.2.10. The agentx-Notify-PDU

An agentx-Notify-PDU is sent by a subagent to cause the master agent to forward a notification.

```
(AgentX header)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| h.version (1) |  h.type (12)  |    h.flags    |  <reserved>   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          h.sessionID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        h.transactionID                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          h.packetID                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        h.payload_length                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

(n.context) OPTIONAL
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Octet String Length (L)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Octet 1      |  Octet 2      |   Octet 3     |   Octet 4     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Octet L - 1  |  Octet L      |        Padding (as required)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

(n.vb)
...
```

An agentx-Notify-PDU contains the following fields:

n.context

An optional non-default context.

n.vb

A VarBindList whose contents define the actual PDU to be sent. This memo places the following restrictions on its contents:

- If the subagent supplies sysUpTime.0, it must be present as the first varbind.

- snmpTrapOID.0 must be present, as the second varbind if sysUpTime.0 was supplied, as the first if it was not.

### 6.2.11 The agentx-Ping-PDU

The agentx-Ping-PDU is sent by a subagent to the master agent to monitor the master agent's ability to receive and send AgentX PDUs over their AgentX session.

```
 (AgentX header)
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | h.version (1) |  h.type (13)  |    h.flags    |  <reserved>   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                          h.sessionID                          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                        h.transactionID                        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                          h.packetID                           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                        h.payload_length                       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

(p.context) OPTIONAL
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                     Octet String Length (L)                   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |  Octet 1      |  Octet 2      |   Octet 3     |   Octet 4     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |  Octet L - 1  |  Octet L      |       Padding (as required)   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

An agentx-Ping-PDU may contain the following field:

      p.context

         An optional non-default context.

   Using p.context a subagent can retrieve the sysUpTime value for a
   specific context, if required.

### 6.2.12.  The agentx-IndexAllocate-PDU

   An agentx-IndexAllocate-PDU is sent by a subagent to request
   allocation of a value for specific index objects.  Refer to section
   7.1.3 (Using the agentx-IndexAllocate-PDU) for suggested usage.

```
    (AgentX header)
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | h.version (1) |  h.type (14)  |    h.flags    |  <reserved>   |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                          h.sessionID                          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                        h.transactionID                        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                          h.packetID                           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                        h.payload_length                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

    (i.context) OPTIONAL
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                     Octet String Length (L)                   |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Octet 1      |  Octet 2      |   Octet 3     |   Octet 4     |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Octet L - 1  |  Octet L      |       Padding (as required)   |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

    (i.vb)
    ...
```

   An agentx-IndexAllocate-PDU contains the following fields:

      i.context

         An optional non-default context.

i.vb

A VarBindList containing the index names and values requested for allocation.

### 6.2.13. The agentx-IndexDeallocate-PDU

An agentx-IndexDeallocate-PDU is sent by a subagent to release previously allocated index values.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| h.version (1) |  h.type (15)  |    h.flags    |  <reserved>   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          h.sessionID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         h.transactionID                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          h.packetID                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        h.payload_length                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

(i.context) OPTIONAL
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Octet String Length (L)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Octet 1      |  Octet 2      |   Octet 3     |   Octet 4     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Octet L - 1  |  Octet L      |       Padding (as required)   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

(i.vb)
...
```

An agentx-IndexDeallocate-PDU contains the following fields:

i.context

An optional non-default context.

i.vb

A VarBindList containing the index names and values to be released.

### 6.2.14. The agentx-AddAgentCaps-PDU

An agentx-AddAgentCaps-PDU is generated by a subagent to inform the master agent of its agent capabilities.

```
(AgentX header)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| h.version (1) |  h.type (16)  |    h.flags    |  <reserved>   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          h.sessionID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        h.transactionID                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          h.packetID                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        h.payload_length                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

(a.context) (OPTIONAL)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Octet String Length (L)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Octet 1      |  Octet 2      |   Octet 3     |   Octet 4     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Octet L - 1  |  Octet L      |        Optional Padding       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

(a.id)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  n_subid      |  prefix       |       0       |  <reserved>   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             sub-identifier #1                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             sub-identifier #n_subid                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

(a.descr)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Octet String Length (L)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Octet 1      |  Octet 2      |   Octet 3     |   Octet 4     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
 ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |  Octet L - 1  |  Octet L      |       Optional Padding        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

An agentx-AddAgentCaps-PDU contains the following fields:

a.context

An optional non-default context.

a.id

An Object Identifier containing the value of an invocation of the AGENT-CAPABILITIES macro, which the master agent exports as a value of sysORID for the indicated context. (Recall that the value of an invocation of an AGENT-CAPABILITIES macro is an object identifier that describes a precise level of support with respect to implemented MIB modules. A more complete discussion of the AGENT-CAPABILITIES macro and related sysORID values can be found in section 6 of RFC 1904 [10].)

a.descr

An Octet String containing a DisplayString to be used as the value of sysORDescr corresponding to the sysORID value above.

### 6.2.15. The agentx-RemoveAgentCaps-PDU

An agentx-RemoveAgentCaps-PDU is generated by a subagent to request that the master agent stop exporting a particular value of sysORID. This value must have previously been advertised by the subagent in an agentx-AddAgentCaps-PDU.

```
(AgentX header)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| h.version (1) |  h.type (17)  |    h.flags    |  <reserved>   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          h.sessionID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        h.transactionID                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          h.packetID                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        h.payload_length                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
(a.context) (OPTIONAL)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Octet String Length (L)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Octet 1      |  Octet 2      |  Octet 3      |  Octet 4      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Octet L - 1  |  Octet L      |       Optional Padding        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

(a.id)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  n_subid      |  prefix       |       0       |  <reserved>   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             sub-identifier #1                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             sub-identifier #n_subid                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

An agentx-RemoveAgentCaps-PDU contains the following fields:

a.context

An optional non-default context.

a.id

An ObjectIdentifier containing the value of sysORID that should no longer be exported.

### 6.2.16. The agentx-Response-PDU

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| h.version (1) |  h.type (18)  |    h.flags    |  <reserved>   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          h.sessionID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        h.transactionID                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           h.packetID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        h.payload_length                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         res.sysUpTime                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           res.error           |     res.index                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...
```

An agentx-Response-PDU contains the following fields:

h.sessionID

If this is a response to a agentx-Open-PDU, then it contains the new and unique sessionID (as assigned by the master agent) for this session.

Otherwise it must be identical to the h.sessionID value in the PDU to which this PDU is a response.

h.transactionID

Must be identical to the h.transactionID value in the PDU to which this PDU is a response.

In an agentx response PDU from the master agent to the subagent, the value of h.transactionID has no significance and can be ignored by the subagent.

h.packetID

Must be identical to the h.packetID value in the PDU to which this PDU is a response.

res.sysUpTime

This field contains the current value of sysUpTime for the indicated context.  It is relevant only in agentx response PDUs sent from the master agent to a subagent in response to the following agentx PDUs:

```
agentx-Open-PDU               (1),
agentx-Close-PDU              (2),
agentx-Register-PDU           (3),
agentx-Unregister-PDU         (4),
agentx-Ping-PDU              (13),
agentx-IndexAllocate-PDU     (14),
agentx-IndexDeallocate-PDU   (15),
agentx-AddAgentCaps-PDU      (16),
agentx-RemoveAgentCaps-PDU   (17)
```

In an agentx response PDU from the subagent to the master agent, the value of res.sysUpTime has no significance and is ignored by the master agent.

res.error

Indicates error status (including `noError'). Values are limited to those defined for errors in the SNMPv2 SMI (RFC 1905 [4]), and the following AgentX-specific values:

```
openFailed                (256),
notOpen                   (257),
indexWrongType            (258),
indexAlreadyAllocated     (259),
indexNoneAvailable        (260),
indexNotAllocated         (261),
unsupportedContext        (262),
duplicateRegistration     (263),
unknownRegistration       (264),
unknownAgentCaps          (265)
```

res.index

In error cases, this is the index of the failed variable binding within a received request PDU. (Note: As explained in section 5.4, Value Representation, the index values of variable bindings within a variable binding list are 1-based.)

A VarBindList may follow these latter two fields, depending on which AgentX PDU is being responded to. These data are specified in the subsequent elements of procedure.

# 7. Elements of Procedure

This section describes the actions of protocol entities (master agents and subagents) implementing the AgentX protocol. Note, however, that it is not intended to constrain the internal architecture of any conformant implementation.

Specific error conditions and associated actions are described in various places. Other error conditions not specifically mentioned fall into one of two categories, "parse" errors and "protocol" errors.

A parse error occurs when a receiving entity cannot decode the PDU. For instance, a VarBind contains an unknown type, or a PDU contains a malformed Object Identifier.

A protocol error occurs when a receiving entity can parse a PDU, but the resulting data is unspecified. For instance, an agentx-Response-PDU is successfully parsed, but contains an unknown res.error value.

An implementation may choose either to ignore such messages, or to close the session on which they are received, using the appropriate reason code as defined in the agentx-Close-PDU.

The actions of AgentX protocol entities can be broadly categorized under two headings, each of which is described separately:

(1) processing AgentX administrative messages (e.g., connection requests from a subagent to a master agent); and

(2) processing SNMP messages (the coordinated actions of a master agent and one or more subagents in processing, for example, a received SNMP GetRequest-PDU).

## 7.1. Processing AgentX Administrative Messages

This subsection describes the actions of AgentX protocol entities in processing AgentX administrative messages. Such messages include those involved in establishing and terminating an AgentX session between a subagent and a master agent, those by which a subagent requests allocation of instance index values, and those by which a subagent communicates to a master agent which MIB regions it supports.

### 7.1.1. Processing the agentx-Open-PDU

When the master agent receives an agentx-Open-PDU, it processes it as follows:

1) An agentx-Response-PDU is created and res.sysUpTime is set to the value of sysUpTime.0 for the indicated context.

2) If the master agent is unable to open an AgentX session for any reason, it may refuse the session establishment request, sending in reply the agentx-Response-PDU, with res.error field set to `openFailed'.

3) Otherwise: The master agent assigns a sessionID to the new session and puts the value in the h.sessionID field of the agentx-Response-PDU. This value must be unique among all existing open sessions.

4) The master agent retains session-specific information from the PDU for this subagent:

   - The NETWORK_BYTE_ORDER value in h.flags is retained. All subsequent AgentX protocol operations initiated by the master agent for this session must use this byte ordering and set this bit accordingly.

     The subagent typically sets this bit to correspond to its native byte ordering, and typically does not vary byte ordering for an initiated session. The master agent must be able to decode each PDU according to the h.flag NETWORK_BYTE_ORDER bit in the PDU, but does not need to toggle its retained value for the session if the subagent varies its byte ordering.

   - The o.timeout value is used in calculating response timeout conditions for this subagent.

   - The o.id and o.descr fields are used for informational purposes. (Such purposes are implementation-specific for now, and may be used in a possible future standard AgentX MIB.)

5) The agentx-Response-PDU is sent with the res.error field set to `noError'.

At this point, an AgentX session is considered established between the master agent and the subagent. An AgentX session is a distinct channel for the exchange of AgentX protocol messages between a master agent and one subagent, qualified by the session-specific attributes listed in 4) above. AgentX session establishment is initiated by the subagent. An AgentX session can be terminated by either the master agent or the subagent.

#### 7.1.2. Processing the agentx-IndexAllocate-PDU

When the master agent receives an agentx-IndexAllocate-PDU, it processes it as follows:

1) An agentx-Response-PDU is created and res.sysUpTime is set to the value of sysUpTime.0 for the default context.

2) If h.sessionID does not correspond to a currently established session with this subagent, the agentx-Response-PDU is sent in reply with res.error set to `notOpen'.

3) If the NON_DEFAULT_CONTEXT bit is set, and the master agent supports only a default context, the agentx-Response-PDU is returned with res.error set to `unsupportedContext', and the requested allocation fails. Otherwise: The value of res.sysUpTime is set to the value of sysUpTime.0 for the indicated context.

4) Each VarBind in the VarBindList is processed until either all are successful, or one fails. If any VarBind fails, the agentx-Response-PDU is sent in reply containing the original VarBindList, with res.index set to indicate the failed VarBind, and with res.error set as described subsequently. All other VarBinds are ignored; no index values are allocated.

   VarBinds are processed as follows:

   - v.name is the name of the index for which a value is to be allocated.

   - v.type is the syntax of the index object.

   - v.data indicates the specific index value requested. If the NEW_INDEX or the ANY_INDEX bit is set, the actual value in v.data is ignored and an appropriate index value is generated.

   a) If there are no currently allocated index values for v.name in the indicated context, and v.type does not correspond to a valid index type value, the VarBind fails and res.error is set to `indexWrongType'.

   b) If there are currently allocated index values for v.name in the indicated context, but the syntax of those values does not match v.type, the VarBind fails and res.error is set to `indexWrongType'.

   c) Otherwise, if both the NEW_INDEX and ANY_INDEX bits are clear, allocation of a specific index value is being requested. If the requested index is already allocated for v.name in the indicated context, the VarBind fails and res.error is set to `indexAlreadyAllocated'.

   d) Otherwise, if the NEW_INDEX bit is set, the master agent should generate the next available index value for v.name in the indicated context, with the constraint that this value must not have been allocated (even if subsequently released) to any subagent since the last re-initialization of the master agent. If no such value can be generated, the VarBind fails and res.error is set to `indexNoneAvailable'.

e) Otherwise, if the ANY_INDEX bit is set, the master agent should generate an index value for v.name in the indicated context, with the constraint that this value is not currently allocated to any subagent. If no such value can be generated, then the VarBind fails and res.error is set to `indexNoneAvailable'.

5) If all VarBinds are processed successfully, the agentx-Response-PDU is sent in reply with res.error set to `noError'. A VarBindList is included that is identical to the one sent in the agentx-IndexAllocate-PDU, except that VarBinds requesting a NEW_INDEX or ANY_INDEX value are generated with an appropriate value.

### 7.1.3. Using the agentx-IndexAllocate-PDU

Index allocation is a service provided by an AgentX master agent. It provides generic support for sharing MIB conceptual tables among subagents who are assumed to have no knowledge of each other.

Each subagent sharing a table should first request allocation of index values, then use those index values to qualify MIB regions in its subsequent registrations.

The master agent maintains a database of index objects (OIDs), and, for each index, the values that have been allocated for it. It is unaware of what MIB variables (if any) the index objects represent.

By convention, subagents use the MIB variable listed in the INDEX clause as the index object for which values must be allocated. For tables indexed by multiple variables, values may be allocated for each index (although this is frequently unnecessary; see example 2 below). The subagent may request allocation of

- a specific index value - an index value that is not currently allocated - an index value that has never been allocated

The last two alternatives reflect the uniqueness and constancy requirements present in many MIB specifications for arbitrary integer indexes (e.g., ifIndex in the IF MIB (RFC 1573 [11]), snmpFddiSMTIndex in the FDDI MIB (RFC 1285 [12]), or sysApplInstallPkgIndex in the System Application MIB [13]). The need for subagents to share tables using such indexes is the main motivation for index allocation in AgentX.

Example 1:

A subagent implements an interface, and wishes to register a single row of the RFC 1573 ifTable. It requests an allocation for the index object "ifIndex", for a value that has never been allocated (since ifIndex values must be unique). The master agent returns the value "7".

The subagent now attempts to register row 7 of ifTable, by specifying a MIB region in the agentx-Register-PDU of 1.3.6.1.2.1.2.2.1.[1-22].7. If the registration succeeds, no further processing is required. The master agent will dispatch to this subagent correctly.

But the registration may fail. Index allocation and MIB region registration are not coupled in the master agent. Some other subagent may have already registered ifTable row 7 without first having requested allocation of the index. The current state of index allocations is not considered when processing registration requests, and the current registry is not considered when processing index allocation requests. If subagents follow the model of "first request allocation of an index, then register the corresponding region", then a successful index allocation request gives a subagent a good hint (but no guarantee) of what it should be able to register.

If the registration failed, the subagent should request allocation of a new index i, and attempt to register ifTable.[1-22].i, until successful.

Example 2:

This same subagent wishes to register ipNetToMediaTable rows corresponding to its interface (ifIndex i). Due to structure of this table, no further index allocation need be done. The subagent can register the MIB region ipNetToMediaTable.[1-4].i, It is claiming responsibility for all rows of the table whose value of ipNetToMediaIfIndex is i.

Example 3:

A network device consists of a set of processors, each of which accepts network connections for a unique set of IP addresses.

Further, each processor contains a subagent that implements tcpConnTable. In order to represent tcpConnTable for the entire managed device, the subagents need to share tcpConnTable.

In this case, no index allocation need be done at all. Each subagent can register a MIB region of tcpConnTable.[1-5].a.b.c.d, where a.b.c.d represents an unique IP address of the individual processor.

Each subagent is claiming responsibility for the region of tcpConnTable where the value of tcpConnLocalAddress is a.b.c.d.

#### 7.1.4 Processing the agentx-IndexDeallocate-PDU

When the master agent receives an agentx-IndexDeallocate-PDU, it processes it as follows:

1) An agentx-Response-PDU is created and res.sysUpTime is set to the value of sysUpTime.0 for the default context.

2) If h.sessionID does not correspond to a currently established session with this subagent, the agentx-Response-PDU is sent in reply with res.error set to `notOpen'.

3) If the NON_DEFAULT_CONTEXT bit is set, and the master agent supports only a default context, the agentx-Response-PDU is returned with res.error set to `unsupportedContext', and the requested deallocation fails. Otherwise: The value of res.sysUpTime is set to the value of sysUpTime.0 for the indicated context.

4) Each VarBind in the VarBindList is processed until either all are successful, or one fails. If any VarBind fails, the agentx-Response-PDU is sent in reply, containing the original VarBindList, with res.index set to indicate the failed VarBind, and with res.error set as described subsequently. All other VarBinds are ignored; no index values are released.

   VarBinds are processed as follows:

   - v.name is the name of the index for which a value is to be released

   - v.type is the syntax of the index object

   - v.data indicates the specific index value to be released. The NEW_INDEX and ANY_INDEX bits are ignored.

   a) If the index value for the named index is not currently allocated to this subagent, the VarBind fails and res.error is set to `indexNotAllocated'.

5) If all VarBinds are processed successfully, res.error is set to `noError' and the agentx-Response-PDU is sent. A VarBindList is included which is identical to the one sent in the agentx-IndexDeallocate-PDU.

   All released index values are now available, and may be used in response to subsequent allocation requests for ANY_INDEX values for the particular index.

### 7.1.5. Processing the agentx-Register-PDU

When the master agent receives an agentx-Register-PDU, it processes it as follows:

1) An agentx-Response-PDU is created and res.sysUpTime is set to the value of sysUpTime.0 for the default context.

2) If h.sessionID does not correspond to a currently established session with this subagent, the agentx-Response-PDU is sent in reply with res.error set to `notOpen'.

3) If the NON_DEFAULT_CONTEXT bit is set, and the master agent supports only a default context, the agentx-Response-PDU is returned with res.error set to `unsupportedContext', and the requested registration fails. Otherwise: The value of res.sysUpTime is set to the value of sysUpTime.0 for the indicated context.

   Note: Non-default contexts might be added on the fly by the master agent, or the master agent might require such non-default contexts to be pre-configured. The choice is implementation-specific.

4) Characterize the request.

   If r.region (or any of its set of Object Identifiers, if r.range is non-zero) is exactly the same as any currently registered value of r.region (or any of its set of Object Identifiers), this registration is termed a duplicate region.

   If r.region (or any of its set of Object Identifiers, if r.range is non-zero) is a subtree of, or contains, any currently registered value of r.region (or any of its set of Object Identifiers), this registration is termed an overlapping region.

   If the NON_DEFAULT_CONTEXT bit is set, this region is to be logically registered within the context indicated by r.context.

Otherwise this region is to be logically registered within the default context.

A registration that would result in a duplicate region with the same priority and within the same context as that of a current registration is termed a duplicate registration.

5) Otherwise, if this is a duplicate registration, the agentx-Response-PDU is returned with res.error set to `duplicateRegistration', and the requested registration fails.

6) Otherwise, the agentx-Response-PDU is returned with res.error set to `noError'.

The master agent adds this region to its registered OID space for the indicated context, to be considered during the dispatching phase for subsequently received SNMP protocol messages.

Note: The following algorithm describes maintaining a set of OID ranges derived from "splitting" registered regions. The algorithm for operational dispatching is also stated in terms of these OID ranges.

These OID ranges are a useful explanatory device, but are not required for a correct implementation.

- If r.region (R1) is a subtree of a currently registered region (R2), split R2 into 3 new regions (R2a, R2b, and R2c) such that R2b is an exact duplicate of R1. Now remove R2 and add R1, R2a, R2b, and R2c to the master agent's lexicographically ordered set of ranges (the registered OID space). Note: Though newly-added ranges R1 and R2b are identical in terms of the MIB objects they contain, they are registered by different subagents, possibly at different priorities.

  For instance, if subagent S2 registered "ip" (R2 is 1.3.6.1.2.1.4) and subagent S1 subsequently registered "ipNetToMediaTable" (R1 is 1.3.6.1.2.1.4.22), the resulting set of registered regions would be:

```
1.3.6.1.2.1.4    up to but not including 1.3.6.1.2.1.4.22     (by S2)
1.3.6.1.2.1.4.22 up to but not including 1.3.6.1.2.1.4.23     (by S2)
1.3.6.1.2.1.4.22 up to but not including 1.3.6.1.2.1.4.23     (by S1)
1.3.6.1.2.1.4.23 up to but not including 1.3.6.1.2.1.5        (by S2)
```

- If r.region (R1) overlaps one or more currently registered regions, then for each overlapped region (R2) split R1 into 3 new ranges (R1a, R1b, R1c) such that R1b is an exact duplicate of R2. Add R1b and R2 into the lexicographically ordered set of regions. Apply (5) above iteratively to R1a and R1c (since they may overlap, or be subtrees of, other regions).

  For instance, given the currently registered regions in the example above, if subagent S3 now registers mib-2 (R1 is 1.3.6.1.2.1) the resulting set of regions would be:

```
1.3.6.1.2.1      up to but not including 1.3.6.1.2.1.4       (by S3)
1.3.6.1.2.1.4    up to but not including 1.3.6.1.2.1.4.22    (by S2)
1.3.6.1.2.1.4    up to but not including 1.3.6.1.2.1.4.22    (by S3)
1.3.6.1.2.1.4.22 up to but not including 1.3.6.1.2.1.4.23    (by S2)
1.3.6.1.2.1.4.22 up to but not including 1.3.6.1.2.1.4.23    (by S1)
1.3.6.1.2.1.4.22 up to but not including 1.3.6.1.2.1.4.23    (by S3)
1.3.6.1.2.1.4.23 up to but not including 1.3.6.1.2.1.5       (by S2)
1.3.6.1.2.1.4.23 up to but not including 1.3.6.1.2.1.5       (by S3)
1.3.6.1.2.1.5    up to but not including 1.3.6.1.2.2         (by S3)
```

Note that at registration time a region may be split into multiple OID ranges due to pre-existing registrations, or as a result of any subsequent registration. This region splitting is transparent to subagents. Hence the master agent must always be able to associate any OID range with the information contained in its original agentx-Register-PDU.

### 7.1.5.1. Handling Duplicate OID Ranges

As a result of this registration algorithm there are likely to be duplicate OID ranges (regions of identical MIB objects registered to different subagents) in the master agent's registered OID space. Whenever the master agent's dispatching algorithm (see 7.2.1, Dispatching AgentX PDUs) results in a duplicate OID range, the master agent selects one to use, termed the 'authoritative region', as follows:

1) Choose the one whose original agentx-Register-PDU r.region contained the most subids, i.e., the most specific r.region. Note: The presence or absence of a range subid has no bearing on how "specific" one object identifier is compared to another.

2) If still ambiguous, there were duplicate regions. Choose the one whose original agentx-Register-PDU specified the smaller value of r.priority.

### 7.1.6. Processing the agentx-Unregister-PDU

1) An agentx-Response-PDU is created and res.sysUpTime is set to the value of sysUpTime.0 for the default context.

2) If h.sessionID does not correspond to a currently established session with this subagent, the agentx-Response-PDU is sent in reply with res.error set to `notOpen'.

3) If the NON_DEFAULT_CONTEXT bit is set, and the master agent supports only a default context, the agentx-Response-PDU is returned with res.error set to `unsupportedContext', and the requested unregistration fails. Otherwise: The value of res.sysUpTime is set to the value of sysUpTime.0 for the indicated context.

4) If u.region, u.priority, and the indicated context do not match an existing registration made during this session, the agentx-Response-PDU is returned with res.error set to `unknownRegistration'.

5) Otherwise, the agentx-Response-PDU is sent in reply with res.error set to `noError', and the previous registration is removed:

   - The master agent removes u.region from its registered OID space within the indicated context. If the original region had been split, all such related regions are removed.

     For instance, given the example registry above, if subagent S2 unregisters "ip", the resulting registry would be:

```
1.3.6.1.2.1      up to but not including 1.3.6.1.2.1.4       (by S3)
1.3.6.1.2.1.4    up to but not including 1.3.6.1.2.1.4.22    (by S3)
1.3.6.1.2.1.4.22 up to but not including 1.3.6.1.2.1.4.23    (by S1)
1.3.6.1.2.1.4.22 up to but not including 1.3.6.1.2.1.4.23    (by S3)
1.3.6.1.2.1.4.23 up to but not including 1.3.6.1.2.1.5       (by S3)
1.3.6.1.2.1.5    up to but not including 1.3.6.1.2.2         (by S3)
```

### 7.1.7. Processing the agentx-AddAgentCaps-PDU

When the master agent receives an agentx-AddAgentCaps-PDU, it processes it as follows:

1) An agentx-Response-PDU is created and res.sysUpTime is set to the value of sysUpTime.0 for the default context.

2) If h.sessionID does not correspond to a currently established session with this subagent, the agentx-Response-PDU is sent in reply with res.error set to `notOpen'.

3) If the NON_DEFAULT_CONTEXT bit is set, and the master agent supports only a default context, the agentx-Response-PDU is returned with res.error set to `unsupportedContext', and the requested operation fails. Otherwise: The value of res.sysUpTime is set to the value of sysUpTime.0 for the indicated context.

4) Otherwise, the master agent adds the subagent's capabilities information to the sysORTable for the indicated context. An agentx-Response-PDU is sent in reply with res.error set to `noError'.

### 7.1.8. Processing the agentx-RemoveAgentCaps-PDU

1) An agentx-Response-PDU is created and res.sysUpTime is set to the value of sysUpTime.0 for the default context.

2) If h.sessionID does not correspond to a currently established session with this subagent, the agentx-Response-PDU is sent in reply with res.error set to `notOpen'.

3) If the NON_DEFAULT_CONTEXT bit is set, and the master agent supports only a default context, the agentx-Response-PDU is returned with res.error set to `unsupportedContext', and the requested operation fails. Otherwise: The value of res.sysUpTime is set to the value of sysUpTime.0 for the indicated context.

4) If the combination of a.id and the optional a.context does not represent a sysORTable entry that was added by this subagent, during this session, the agentx-Response-PDU is returned with res.error set to `unknownAgentCaps'.

5) Otherwise the master agent deletes the corresponding sysORTable entry and sends in reply the agentx-Response-PDU, with res.error set to `noError'.

### 7.1.9. Processing the agentx-Close-PDU

When the master agent receives an agentx-Close-PDU, it processes it as follows:

1) An agentx-Response-PDU is created and res.sysUpTime is set to the value of sysUpTime.0 for the default context.

2) If h.sessionID does not correspond to a currently established session with this subagent, the agentx-Response-PDU is sent in reply with res.error set to `notOpen'.

3) Otherwise, the master agent closes the AgentX session as described below. No agentx-Response-PDU is sent.

   - All MIB regions that have been registered during this session are unregistered, as described in 7.1.6.

   - All index values allocated during this session are freed, as described in section 7.1.4.

   - All sysORID values that were registered during this session are removed, as described in section 7.1.8.

The master agent does not maintain state for closed sessions. If a subagent wishes to re-establish a session after receiving an agentx-Close-PDU, it needs to re-register MIB regions, agent capabilities, etc.

### 7.1.10. Detecting Connection Loss

If a master agent is able to detect (from the underlying transport) that a subagent cannot receive AgentX PDUs, it should close all affected AgentX sessions as described in 7.1.9, step 3).

### 7.1.11. Processing the agentx-Notify-PDU

A subagent sending SNMPv1 trap information must map this into (minimally) a value of snmpTrapOID.0, as described in 3.1.2 of RFC 1908 [8].

The master agent processes the agentx-Notify-PDU as follows:

1) If h.sessionID does not correspond to a currently established session with this subagent, an agentx-Response-PDU is sent in reply with res.error set to `notOpen', and res.sysUpTime set to the value of sysUpTime.0 for the indicated context.

2) The VarBindList is parsed. If it does not contain a value for sysUpTime.0, the master agent supplies the current value of sysUpTime.0 for the indicated context. If the next VarBind (either the first or second VarBind; see section 6.2.10.1) is not snmpTrapOID.0, the master agent ceases further processing of the notification.

3) Notifications are sent according to the implementation-specific configuration of the master agent.

   If SNMPv1 Trap PDUs are generated, the recommended mapping is as described in RFC 2089 [9].

   Except in the case of a `notOpen' error as described in (1) above, no agentx-Response-PDU is sent to the subagent when the master agent finishes processing the notification.

### 7.1.12. Processing the agentx-Ping-PDU

When the master agent receives an agentx-Ping-PDU, it processes it as follows:

1) An agentx-Response-PDU is created and res.sysUpTime is set to the value of sysUpTime.0 for the default context.

2) If h.sessionID does not correspond to a currently established session with this subagent, the agentx-Response-PDU is sent in reply with res.error set to `notOpen'.

3) If the NON_DEFAULT_CONTEXT bit is set, and the master agent supports only a default context, the agentx-Response-PDU is returned with res.error set to `unsupportedContext'. Otherwise: The value of res.sysUpTime is set to the value of sysUpTime.0 for the indicated context.

4) The agentx-Response-PDU is sent, with res.error set to `noError'.

If a subagent does not receive a response to its pings, or if it is able to detect (from the underlying transport) that the master agent is not able to receive AgentX messages, then it eventually must initiate a new AgentX session, re-register its regions, etc.

### 7.2. Processing Received SNMP Protocol Messages

When an SNMP GetRequest, GetNextRequest, GetBulkRequest, or SetRequest protocol message is received by the master agent, the master agent applies its access control policy.

In particular, for SNMPv1 or SNMPv2c PDUs, the master agent applies the Elements of Procedure defined in section 4.1 of RFC 1157 [6] that apply to receiving entities. (For other versions of SNMP, the master agent applies the access control policy defined in the Elements of Procedure for those versions.)

In the SNMPv1 or v2c frameworks, the master agent uses the community string as an index into a local repository of configuration information that may include community profiles or more complex context information.

If application of the access control policy results in a valid SNMP request PDU, then an SNMP Response-PDU is constructed from information gathered in the exchange of AgentX PDUs between the master agent and one or more subagents. Upon receipt and initial validation of an SNMP request PDU, a master agent uses the procedures described below to dispatch AgentX PDUs to the proper subagents, marshal the subagent responses, and construct an SNMP response PDU.

### 7.2.1. Dispatching AgentX PDUs

Upon receipt and initial validation of an SNMP request PDU, a master agent uses the procedures described below to dispatch AgentX PDUs to the proper subagents.

Note: In the following procedures, an object identifier is said to be "contained" within an OID range when both of the following are true:

- The object identifier does not lexicographically precede the range.

- The object identifier lexicographically precedes the end of the range.

General Rules of Procedure

While processing a particular SNMP request, the master agent may send one or more AgentX PDUs to one or more subagents. The following rules of procedure apply in general to the AgentX master agent. PDU-specific rules are listed in the applicable sections.

1) Honoring the registry

   Because AgentX supports overlapping registrations, it is possible for the master agent to obtain a value for a requested varbind from within multiple registered MIB regions.

   The master agent must ensure that the value (or exception) actually returned in the SNMP response PDU is taken from the authoritative region (as defined in section 7.1.5.1).

2) GetNext and GetBulk Processing

The master agent may choose to send agentx-Get-PDUs while servicing an SNMP GetNextRequest-PDU. The master agent may choose to send agentx-Get-PDUs or agentx-GetNext-PDUs while servicing an SNMP GetBulkRequest-PDU. One possible reason for this would be if the current iteration has targeted instance-level registrations.

The master agent may choose to "scope" the possible instances returned by a subagent by specifying an ending OID in the SearchRange. If such scoping is used, typically the ending OID would be the first lexicographical successor to the target OID range that was registered by a subagent other than the target subagent. Regardless of this choice, rule (1) must be obeyed.

The master agent may require multiple request-response iterations on the same subagent session, to determine the final value of all requested variables.

All AgentX PDUs sent on the session while processing a given SNMP request must contain identical values of transactionID. Each different SNMP request processed by the master agent must present a unique value of transactionID (within the limits of the 32-bit field) to the session.

3) Number and order of variables sent per AgentX PDU

For Get/GetNext/GetBulk operations, at any stage of the possibly iterative process, the master agent may need to dispatch several SearchRanges to a particular subagent session. The master agent may send one, some, or all of the SearchRanges in a single AgentX PDU.

The master agent must ensure that the correct contents and ordering of the VarBindList in the SNMP Response-PDU are maintained.

The following rules govern the number of VarBinds in a given AgentX PDU:

a) The subagent must support processing of AgentX PDUs with multiple VarBinds.

b) When processing an SNMP Set request, the master agent must send all of the VarBinds applicable to a particular subagent session in a single Test/Set transaction.

c) When processing an SNMP Get, GetNext, or GetBulk request, the master agent may send a single AgentX PDU to the subagent with all applicable VarBinds, or multiple PDUs with single VarBinds, or something in between those extremes. The determination of which method to use in a particular case is implementation-specific.

4) Timeout Values

The master agent chooses a timeout value for each MIB region being queried, which is

a) the value specified during registration of the MIB region, if it was non-zero

b) otherwise, the value specified during establishment of the session in which this region was subsequently registered, if that value was non-zero.

c) otherwise, the master agent's default value

When an AgentX PDU that references multiple MIB regions is dispatched, the timeout value used for the PDU is the maximum value of the timeouts so determined for each of the referenced MIB regions.

5) Context

If the master agent has determined that a specific non-default context is associated with the SNMP request PDU, that context is encoded into the AgentX PDU's context field and the NON_DEFAULT_CONTEXT bit is set in h.flags.

Otherwise, no context Octet String is added to the PDU, and the NON_DEFAULT_CONTEXT bit is cleared.

#### 7.2.1.1. agentx-Get-PDU

Each variable binding in the SNMP request PDU is processed as follows:

(1) Identify the target OID range.

Within a lexicographically ordered set of OID ranges, valid for the indicated context, locate the authoritative region that contains the binding's name.

(2) If no such OID range exists, the variable binding is not processed further, and its value is set to `noSuchObject'.

(3) Identify the subagent session in which this region was registered, termed the target session.

(4) If this is the first variable binding to be dispatched over the target session in a request-response exchange entailed in the processing of this management request:

- Create an agentx-Get-PDU for this session, with the header fields initialized as described above (see 6.1 AgentX PDU Header).

(5) Add a SearchRange to the end of the target session's PDU for this variable binding.

- The variable binding's name is encoded into the starting OID.

- The ending OID is encoded as null.

### 7.2.1.2. agentx-GetNext-PDU

Each variable binding in the SNMP request PDU is processed as follows:

(1) Identify the target OID range.

Within a lexicographically ordered set of OID ranges, valid for the indicated context, locate

a) the authoritative OID range that contains the variable binding's name and is not a fully qualified instance, or

b) the authoritative OID range that is the first lexicographical successor to the variable binding's name.

(2) If no such OID range exists, the variable binding is not processed further, and its value is set to `endOfMibView'.

(3) Identify the subagent session in which this region was registered, termed the target session.

(4) If this is the first variable binding to be dispatched over the target session in a request-response exchange entailed in the processing of this management request:

- Create an agentx-GetNext-PDU for the session, with the header fields initialized as described above (see 6.1 AgentX PDU Header).

(5) Add a SearchRange to the end of the target session's agentx-GetNext-PDU for this variable binding.

  - if (1a) applies, the variable binding's name is encoded into the starting OID, and the OID's "include" field is set to 0.

  - if (1b) applies, the target OID is encoded into the starting OID, and its "include" field is set to 1.

#### 7.2.1.3. agentx-GetBulk-PDU

(Note: The outline of the following procedure is based closely on section 4.2.3, "The GetBulkRequest-PDU" of RFC 1905 [4]. Please refer to it for details on the format of the SNMP GetBulkRequest-PDU itself.)

Each variable binding in the request PDU is processed as follows:

(1) Identify the authoritative target OID range and target session, exactly as described for the agentx-GetNext-PDU (see 7.2.1.2).

(2) If this is the first variable binding to be dispatched over the target session in a request-response exchange entailed in the processing of this management request:

- Create an agentx-GetBulk-PDU for the session, with the header fields initialized as described above (see 6.1 AgentX PDU Header).

(3) Add a SearchRange to the end of the target session's agentx-GetBulk-PDU for this variable binding, as described for the agentx-GetNext-PDU. If the variable binding was a non-repeater in the original request PDU, it must be a non-repeater in the agentx-GetBulk-PDU.

The value of g.max_repetitions in the agentx-GetBulk-PDU may be less than (but not greater than) the value in the original request PDU.

The master agent may make such alterations due to simple sanity checking, optimizations for the current iteration based on the registry, the maximum possible size of a potential Response-PDU, known constraints of the AgentX transport, or any other implementation-specific constraint.

### 7.2.1.4. agentx-TestSet-PDU

AgentX employs test-commit-undo-cleanup phases to achieve "as if simultaneous" semantics of the SNMP SetRequest-PDU within the extensible agent. The initial phase involves the agentx-TestSet-PDU.

Each variable binding in the SNMP request PDU is processed in order, as follows:

(1) Identify the target OID range.

Within a lexicographically ordered set of OID ranges, valid for the indicated context, locate the authoritative range that contains the variable binding's name.

(2) If no such OID range exists, this variable binding fails with an error of `notWritable'. Processing is complete for this request.

(3) Identify the single subagent responsible for this OID range, termed the target subagent, and the applicable session, termed the target session.

(4) If this is the first variable binding to be dispatched over the target session in a request-response exchange entailed in the processing of this management request:

- create an agentx-TestSet-PDU for the session, with the header fields initialized as described above (see 6.1 AgentX PDU Header).

(5) Add a VarBind to the end of the target session's PDU for this variable binding, as described in section 5.4.

Note that all VarBinds applicable to a given session must be sent in a single agentx-TestSet-PDU.

### 7.2.1.5. Dispatch

A timeout value is calculated for each PDU to be sent, which is the maximum value of the timeouts determined for each of the PDU's SearchRanges (as described above in 7.2.1 Dispatching AgentX PDUs, item 4). Each pending PDU is mapped (via its h.sessionID value) to a particular transport domain/endpoint, as described in section 8 (Transport Mappings).

### 7.2.2. Subagent Processing of agentx-Get, GetNext, GetBulk-PDUs

A conformant AgentX subagent must support the agentx-Get, -GetNext, and -GetBulk PDUs, and must support multiple variables being supplied in each PDU.

When a subagent receives an agentx-Get-, GetNext-, or GetBulk-PDU, it performs the indicated management operations and returns an agentx-Response-PDU.

The agentx-Response-PDU header fields are identical to the received request PDU except that, at the start of processing, the subagent initializes h.type to Response, res.error to `noError', res.index to 0, and the VarBindList to null.

Each SearchRange in the request PDU's SearchRangeList is processed as described below, and a VarBind is added in the corresponding location of the agentx-Response-PDU's VarbindList. If processing should fail for any reason not described below, res.error is set to `genErr', res.index to the index of the failed SearchRange, the VarBindList is reset to null, and this agentx-Response-PDU is returned to the master agent.

#### 7.2.2.1. Subagent Processing of the agentx-Get-PDU

Upon the subagent's receipt of an agentx-Get-PDU, each SearchRange in the request is processed as follows:

(1) The starting OID is copied to v.name.

(2) If the starting OID exactly matches the name of a variable instantiated by this subagent within the indicated context and session, v.type and v.data are encoded to represent the variable's syntax and value, as described in section 5.4, Value Representation.

(3) Otherwise, if the starting OID does not match the object identifier prefix of any variable instantiated within the indicated context and session, the VarBind is set to `noSuchObject', in the manner described in section 5.4, Value Representation.

(4) Otherwise, the VarBind is set to `noSuchInstance' in the manner described in section 5.4, Value Representation.

#### 7.2.2.2. Subagent Processing of the agentx-GetNext-PDU

Upon the subagent's receipt of an agentx-GetNext-PDU, each SearchRange in the request is processed as follows:

(1) The subagent searches for a variable within the lexicographically ordered list of variable names for all variables it instantiates (without regard to registration of regions) within the indicated context and session, for which the following are all true:

- if the "include" field of the starting OID is 0, the variable's name is the closest lexicographical successor to the starting OID.

- if the "include" field of the starting OID is 1, the variable's name is either equal to, or the closest lexicographical successor to, the starting OID.

- If the ending OID is not null, the variable's name lexicographically precedes the ending OID.

If all of these conditions are met, v.name is set to the located variable's name. v.type and v.data are encoded to represent the variable's syntax and value, as described in section 5.4, Value Representation.

(2) If no such variable exists, v.name is set to the starting OID, and the VarBind is set to `endOfMibView', in the manner described in section 5.4, Value Representation.

#### 7.2.2.3. Subagent Processing of the agentx-GetBulk-PDU

A maximum of N + (M * R) VarBinds are returned, where

N equals g.non_repeaters,
M equals g.max_repetitions, and
R is (number of SearchRanges in the GetBulk request) - N.

The first N SearchRanges are processed exactly as for the agentx-GetNext-PDU.

If M and R are both non-zero, the remaining R SearchRanges are processed iteratively to produce potentially many VarBinds. For each iteration i, such that i is greater than zero and less than or equal to M, and for each repeated SearchRange s, such that s is greater than zero and less than or equal to R, the (N+((i-1)*R)+s)-th VarBind is added to the agentx-Response-PDU as follows:

1) The subagent searches for a variable within the lexicographically ordered list of variable names for all variables it instantiates (without regard to registration of regions) within the indicated context and session, for which the following are all true:

   - The variable's name is the (i)-th lexicographical successor to the (N+s)-th requested OID.

     (Note that if i is 0 and the "include" field is 1, the variable's name may be equivalent to, or the first lexicographical successor to, the (N+s)-th requested OID.)

   - If the ending OID is not null, the variable's name lexicographically precedes the ending OID.

   If all of these conditions are met, v.name is set to the located variable's name. v.type and v.data are encoded to represent the variable's syntax and value, as described in section 5.4, Value Representation.

2) If no such variable exists, the VarBind is set to `endOfMibView' as described in section 5.4, Value Representation. v.name is set to v.name of the (N+((i-2)*R)+s)-th VarBind unless i is currently 1, in which case it is set to the value of the starting OID in the (N+s)-th SearchRange.

Note that further iterative processing should stop if

- For any iteration i, all s values of v.type are `endOfMibView'.

- An AgentX transport constraint or other implementation-specific constraint is reached.

### 7.2.3. Subagent Processing of agentx-TestSet, -CommitSet, -UndoSet, -CleanupSet-PDUs

A conformant AgentX subagent must support the agentx-TestSet, -CommitSet, -UndoSet, and -CleanupSet PDUs, and must support multiple variables being supplied in each PDU.

These four PDUs are used to collectively perform the indicated management operation. An agentx-Response-PDU is sent in reply to each of the PDUs, to inform the master agent of the state of the operation.

The agentx-Response-PDU header fields are identical to the received request PDU except that, at the start of processing, the subagent initializes h.type to Response, res.error to `noError', and res.index to 0.

These Response-PDUs do not contain a VarBindList.

#### 7.2.3.1. Subagent Processing of the agentx-TestSet-PDU

Upon the subagent's receipt of an agentx-TestSet-PDU, each VarBind in the PDU is validated until they are all successful, or until one fails, as described in section 4.2.5 of RFC 1905 [4]. The subagent validates variables with respect to the context and session indicated in the testSet-PDU.

If each VarBind is successful, the subagent has a further responsibility to ensure the availability of all resources (memory, write access, etc.) required for successfully carrying out a subsequent agentx-CommitSet operation. If this cannot be guaranteed, the subagent should set res.error to `resourceUnavailable'.

As a result of this validation step, an agentx-Response-PDU is sent in reply whose res.error field is set to one of the following (SNMPv2 SMI) values:

```
noError                    (0),
genErr                     (5),
noAccess                   (6),
wrongType                  (7),
wrongLength                (8),
wrongEncoding              (9),
wrongValue                (10),
noCreation                (11),
inconsistentValue         (12),
resourceUnavailable       (13),
notWritable               (17),
inconsistentName          (18)
```

If this value is not `noError', the res.index field must be set to the index of the VarBind for which validation failed.

Implementation of rigorous validation code may be one of the most demanding aspects of subagent development. Implementors are strongly encouraged to do this right, so as to avoid if at all possible the extensible agent's having to return `commitFailed' or `undoFailed' during subsequent processing.

7.2.3.2. Subagent Processing of the agentx-CommitSet-PDU

The agentx-CommitSet-PDU indicates that the subagent should actually perform (as described in the post-validation sections of 4.2.5 of RFC 1905 [4]) the management operation indicated by the previous TestSet-PDU. After carrying out the management operation, the subagent sends in reply an agentx-Response-PDU whose res.error field is set to one of the following (SNMPv2 SMI) values:

```
noError                  (0),
commitFailed            (14)
```

If this value is `commitFailed', the res.index field must be set to the index of the VarBind for which the operation failed. Otherwise res.index is set to 0.

7.2.3.3. Subagent Processing of the agentx-UndoSet-PDU

The agentx-UndoSet-PDU indicates that the subagent should undo the management operation requested in a preceding CommitSet-PDU. The undo process is as described in section 4.2.5 of RFC 1905 [4].

After carrying out the undo process, the subagent sends in reply an agentx-Response-PDU whose res.index field is set to 0, and whose res.error field is set to one of the following (SNMPv2 SMI) values:

```
noError                  (0),
undoFailed              (15)
```

If this value is `undoFailed', the res.index field must be set to the index of the VarBind for which the operation failed. Otherwise res.index is set to 0.

This PDU also signals the end of processing of the management operation initiated by the previous TestSet-PDU. The subagent should release resources, etc. as described in section 7.2.3.4.

7.2.3.4. Subagent Processing of the agentx-CleanupSet-PDU

The agentx-CleanupSet-PDU signals the end of processing of the management operation requested in the previous TestSet-PDU. This is an indication to the subagent that it may now release any resources it may have reserved in order to carry out the management request.

No response is sent by the subagent.

### 7.2.4. Master Agent Processing of AgentX Responses

The master agent now marshals all subagent AgentX response PDUs and builds an SNMP response PDU. In the next several subsections, the initial processing of all subagent AgentX response PDUs is described, followed by descriptions of subsequent processing for each specific subagent Response.

#### 7.2.4.1. Common Processing of All AgentX Response PDUs

1) If a subagent does not respond within the timeout interval for this dispatch, it is treated as if the subagent had returned `genErr' and processed as described below.

   A timeout may be due to a variety of reasons, and does not necessarily denote a failed or malfunctioning subagent. As such, the master agent's response to a subagent timeout is implementation-specific, but with the following constraint:

   > A subagent that times out on three consecutive requests is considered unable to respond, and the master agent must close the AgentX session as described in 7.1.9, step (2).

2) Otherwise, the h.packetID, h.sessionID, and h.transactionID fields of the AgentX response PDU are used to correlate subagent responses. If the response does not pertain to this SNMP operation, it is ignored.

3) Otherwise, the responses are processed jointly to form the SNMP response PDU.

#### 7.2.4.2. Processing of Responses to agentx-Get-PDUs

After common processing of the subagent's response to an agentx-Get-PDU (see 7.2.4.1 above), processing continues with the following steps:

1) For any received AgentX response PDU, if res.error is not `noError', the SNMP response PDU's error code is set to this value, and its error index to the index of the variable binding corresponding to the failed VarBind in the subagent's AgentX response PDU.

   All other AgentX response PDUs received due to processing this SNMP request are ignored. Processing is complete; the SNMP Response PDU is ready to be sent (see section 7.2.5, Sending the SNMP Response-PDU).

2) Otherwise, the content of each VarBind in the AgentX response PDU is used to update the corresponding variable binding in the SNMP Response-PDU.

### 7.2.4.3. Processing of Responses to agentx-GetNext-PDU and agentx-GetBulk-PDU

After common processing of the subagent's response to an agentx-GetNext-PDU or agentx-GetBulk-PDU (see 7.2.4.1 above), processing continues with the following steps:

1) For any received AgentX response PDU, if res.error is not `noError', the SNMP response PDU's error code is set to this value, and its error index to the index of the VarBind corresponding to the failed VarBind in the subagent's AgentX response PDU.

   All other AgentX response PDUs received due to processing this SNMP request are ignored. Processing is complete; the SNMP response PDU is ready to be sent (see section 7.2.5, Sending the SNMP Response PDU).

2) Otherwise, the content of each VarBind in the AgentX response PDU is used to update the corresponding VarBind in the SNMP response PDU.

After all expected AgentX response PDUs have been processed, if any VarBinds still contain the value `endOfMibView' in their v.type fields, processing must continue:

3) A new iteration of AgentX request dispatching is initiated (as described in section 7.2.1.1), in which only those VarBinds whose v.type is `endOfMibView' are processed.

4) For each such VarBind, a target OID range is identified which is the lexicographical successor to the target OID range for this VarBind on the last iteration. The target subagent is the one that registered the target OID range. The target session is the one in which the target OID range was registered.

   If an agentx-GetNext- or GetBulk-PDU is being dispatched, the starting OID in the SearchRanges is set to the target OID range, and its "include" field is set to 1.

5) The value of transactionID must be identical to the value used during the previous iteration.

6) The AgentX PDUs are sent to the subagent(s), and the responses are received and processed according to the steps described in section 7.2.4.

7) This process continues iteratively until a complete SNMP Response-PDU has been built, or until there remain no target OID range lexicographical successors.

#### 7.2.4.4. Processing of Responses to agentx-TestSet-PDUs

After common processing of the subagent's response to an agentx-TestSet-PDU (see 7.2.4.1 above), processing continues with the further exchange of AgentX PDUs. The value of h.transactionID in the agentx-CommitSet, -UndoSet, and -CleanupSet-PDUs must be identical to the value sent in the testSet-PDU.

The state transitions and PDU sequences are depicted in section 7.3.

1) If any target subagent's response is not `noError', all other agentx-Response-PDUs received due to processing this SNMP request are ignored.

   An agentx-CleanupSet-PDU is sent to each target subagent that has been sent a agentx-TestSet-PDU.

   Processing is complete; the SNMP response PDU is constructed as described below in 7.2.4.6.

2) Otherwise an agentx-CommitSet-PDU is sent to each target subagent.

#### 7.2.4.5. Processing of Responses to agentx-CommitSet-PDUs

After common processing of the subagent's response to an agentx-CommitSet-PDU (see 7.2.4.1 above), processing continues with the following steps:

1) If any response is not `noError', all other agentx-Response-PDUs received due to processing this SNMP request are ignored.

   An agentx-UndoSet-PDU is sent to each target subagent that has been sent a agentx-CommitSet-PDU. All other subagents are sent a agentx-CleanupSet-PDU.

2) Otherwise an agentx-CleanupSet-PDU is sent to each target subagent. Processing is complete; the SNMP response PDU is constructed as described below in 7.2.4.6.

#### 7.2.4.6. Processing of Responses to agentx-UndoSet-PDUs

After common processing of the subagent's response to an agentx-UndoSet-PDU (see 7.2.4.1 above), processing continues with the following steps:

1) If any response is not `noError' the SNMP response PDU's error code is set to this value, and its error index to the index of the VarBind corresponding to the failed VarBind in the agentx-TestSet-PDU.

   Otherwise the SNMP response PDU's error code is set to `noError' and its error index to 0.

### 7.2.5. Sending the SNMP Response-PDU

Once the processing described in sections 7.2.1 - 7.2.4 is complete, there is an SNMP response PDU available. The master agent now implements the Elements of Procedure for the applicable version of the SNMP protocol in order to encapsulate the PDU into a message, and transmit it to the originator of the SNMP management request. Note that this may involve altering the PDU contents (for instance, to replace the original VarBinds if an error condition is to be returned).

The response PDU may also be altered in order to support the SNMP version 1 framework. In such cases the required mapping is that defined in RFC 2089 [9]. (Note in particular that the rules for handling Counter64 syntax may require re-sending AgentX GetBulk or GetNext PDUs until a VarBind of suitable syntax is returned.)

### 7.2.6. MIB Views

AgentX subagents are not aware of MIB views, since view information is not contained in AgentX PDUs.

As stated above, the descriptions of procedures in section 7 of this memo are not intended to constrain the internal architecture of any conformant implementation. In particular, the master agent procedures described in sections 7.2.1 and 7.2.4 may be altered so as to optimize AgentX exchanges when implementing MIB views.

Such optimizations are beyond the scope of this memo. But note that section 7.2.3 defines subagent behavior in such a way that alteration of SearchRanges may be used in such optimizations.

## 7.3. State Transitions

State diagrams are presented from the master agent's perspective for transport connection and session establishment, and from the subagent's perspective for Set transaction processing.

### 7.3.1. Set Transaction States

The following table presents, from the subagent's perspective, the state transitions involved in Set transaction processing:

| EVENT \ STATE | A (Initial State) | B TestOK | C Commit OK | D Test Fail | E Commit Fail |
|---|---|---|---|---|---|
| Receive TestSet PDU | 7.2.3.1 All varbinds OK? Yes ->B No ->D | X | X | X | X |
| Receive Commit-Set PDU | X | 7.2.3.2 NoError? Yes ->C No ->E | X | X | X |
| Receive UndoSet PDU | X | X | 7.2.3.3 ->done | X | 7.2.4.5 ->done |
| Receive Cleanup-Set PDU | X | 7.2.4.4 ->done | 7.2.3.4 ->done | 7.2.4.4 ->done | X |
| Session Loss | ->done | rollback ->done | undo ->done | ->done | ->done |

There are three possible sequences that a subagent may follow for a particular set transaction:

1) TestSet CommitSet CleanupSet
2) TestSet CommitSet UndoSet
3) TestSet CleanupSet

Note that a single PDU sequence may result in multiple paths through the finite state machine (FSM). For example, the sequence

```
TestSet CommitSet UndoSet
```

may walk through either of these two state sequences:

```
(initial) TestOK CommitOK   (done)
(initial) TestOK CommitFail (done)
```

### 7.3.2 Transport Connection States

The following table presents, from the master agent's perspective, the state transitions involved in transport connection setup and teardown:

| EVENT | STATE: A No transport | STATE: B Transport connected |
| --- | --- | --- |
| Transport connect indication | ->B | X |
| Receive Open-PDU | X | if duplicate session id, reject, else establish session<br>->B |
| Receive Response-PDU | X | if matching session id, feed to that session's FSM else ignore<br>->B |
| Receive other PDUs | X | if matching session id, feed to that session's FSM else reject<br>->B |
| Transport disconnect indication | X | notify all sessions on this transport<br>->A |

### 7.3.3 Session States

The following table presents, from the master agent's perspective, the state transitions involved in session setup and teardown:

| EVENT | STATE: A No session | STATE: B Session established |
|---|---|---|
| Receive Open PDU | 7.1.1 ->B | X |
| Receive Close PDU | X | 7.1.9 ->A |
| Receive Register PDU | X | 7.1.5 ->B |
| Receive Unregister PDU | X | 7.1.6 ->B |
| Receive Get PDU GetNext PDU GetBulk PDU TestSet PDU CommitSet PDU UndoSet PDU CleanupSet PDU | X | X |
| Receive Notify PDU | X | 7.1.11 ->B |
| Receive Ping PDU | X | 7.1.12 ->B |

(continued next page)

| | | |
|---|---|---|
| Receive IndexAllocate PDU | X | 7.1.2 ->B |
| Receive IndexDeallocate PDU | X | 7.1.4 ->B |
| Receive AddAgentxCaps PDU | X | 7.1.7 ->B |
| Receive RemoveAgentxCap PDU | X | 7.1.8 ->B |
| Receive Response PDU | X | 7.2.4 ->B |
| Receive Other PDU | X | X |

## 8. Transport Mappings

The same AgentX PDU formats, encodings, and elements of procedure are used regardless of the underlying transport.

### 8.1. AgentX over TCP

#### 8.1.1. Well-known Values

The master agent accepts TCP connection requests for the well-known port 705. Subagents connect to the master agent using this port number.

#### 8.1.2. Operation

Once a TCP connection has been established, the AgentX peers use this connection to carry all AgentX PDUs. Multiple AgentX sessions may be established using the same TCP connection. AgentX PDUs are sent within an AgentX session. AgentX peers are responsible for mapping the h.sessionID to a particular TCP connection.

All AgentX PDUs are presented individually to the TCP, to be sent as the data portion of a TCP PDU.

### 8.2. AgentX over UNIX-domain Sockets

Many (BSD-derived) implementations of the UNIX operating system support the UNIX pathname address family (AF_UNIX) for socket communications. This provides a convenient method of sending and receiving data between processes on the same host.

Mapping AgentX to this transport is useful for environments that

- wish to guarantee subagents are running on the same managed node as the master agent, and where

- sockets provide better performance than TCP or UDP, especially in the presence of heavy network I/O

#### 8.2.1. Well-known Values

The master agent creates a well-known UNIX-domain socket endpoint called "/var/agentx/master". (It may create other, implementation-specific endpoints.)

This endpoint name uses the character set encoding native to the managed node, and represents a UNIX-domain stream (SOCK_STREAM) socket.

#### 8.2.2. Operation

Once a connection has been established, the AgentX peers use this connection to carry all AgentX PDUs.

Multiple AgentX sessions may be established using the same connection. AgentX PDUs are sent within an AgentX session. AgentX peers are responsible for mapping the h.sessionID to a particular connection.

All AgentX PDUs are presented individually to the socket layer, to be sent in the data stream.

## 9. Security Considerations

This memo defines a protocol between two processing entities, one of which (the master agent) is assumed to perform authentication of received SNMP requests and to control access to management information. The master agent performs these security operations independently of the other processing entity (the subagent).

Security considerations require three questions to be answered:

1. Is a particular subagent allowed to initiate a session with a particular master agent?

2. During an AgentX session, is any SNMP security-related information (for example, community names) passed from the master agent to the subagent?

3. During an AgentX session, what part of the MIB tree is this subagent allowed to register?

The answer to the third question is: A subagent can register any subtree (subject to AgentX elements of procedure, section 7.1.5). Currently there is no access control mechanism defined in AgentX. A concern here is that a malicious subagent that registers an unauthorized "sensitive" subtree, could see modification requests to those objects, or by giving its own clever answer to NMS queries, could cause the NMS to do something that leads to information disclosure or other damage.

The answer to the second question is: No.

Now we can answer the first question. AgentX does not contain a mechanism for authorizing/refusing session initiations. Thus, controlling subagent access to the master agent may only be done at a lower layer (e.g., transport).

An AgentX subagent can connect to a master agent using either a network transport mechanism (e.g., TCP), or a "local" mechanism (e.g., shared memory, named pipes).

In the case where a local transport mechanism is used and both subagent and master agent are running on the same host, connection authorization can be delegated to the operating system features. The answer to the first security question then becomes: "If and only if the subagent has sufficient privileges, then the operating system will allow the connection".

If a network transport is used, currently there is no inherent security. Transport Layer Security or SSL could be used to control subagent connections, but that is beyond the scope of this document.

Thus it is recommended that subagents always run on the same host as the master agent and that operating system features be used to ensure that only properly authorized subagents can establish connections to the master agent.

## 10. Acknowledgements

The initial development of this memo was heavily influenced by the DPI 2.0 specification RFC 1592 [7].

This document was produced by the IETF Agent Extensibility (AgentX) Working Group, and benefited especially from the contributions of the following working group members:

David Battle, Uri Blumenthal, Jeff Case, Maria Greene, Dave Keeney, Harmen van der Linde, Bob Natale, Randy Presuhn, Aleksey Romanov, Don Ryan, and Juergen Schoenwaelder.

The AgentX Working Group is chaired by:

Bob Natale
ACE*COMM Corporation
704 Quince Orchard Road
Gaithersburg MD  20878

Phone: +1-301-721-3000
Fax:   +1-301-721-3001
EMail: bnatale@acecomm.com

## 11. Authors' and Editor's Addresses

Mike Daniele
Digital Equipment Corporation
110 Spit Brook Rd
Nashua, NH 03062

Phone: +1-603-881-1423
EMail: daniele@zk3.dec.com

Bert Wijnen
IBM Professional Services
Watsonweg 2
1423 ND Uithoorn
The Netherlands

Phone: +31-79-322-8316
EMail: wijnen@vnet.ibm.com

Dale Francisco (editor)
Cisco Systems
150 Castilian Dr
Goleta CA 93117

Phone: +1-805-961-3642
Fax: +1-805-961-3600
EMail: dfrancis@cisco.com

## 12. References

[1] Information processing systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1), International Organization for Standardization. International Standard 8824, (December, 1987).

[2] 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.

[3] Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1903, January 1996.

[4] 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.

[5] 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.

[6] Case, J., Fedor, M., Schoffstall, M., and J. Davin, "Simple Network Management Protocol", STD 15, RFC 1157, SNMP Research, Performance Systems International, MIT Laboratory for Computer Science, May 1990.

[7] Wijnen, B., Carpenter, G., Curran, K., Sehgal, A. and G. Waters, "Simple Network Management Protocol: Distributed Protocol Interface, Version 2.0", RFC 1592, March 1994.

[8] 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.

[9] Wijnen, B. and D. Levi, "V2ToV1: Mapping SNMPv2 onto SNMPv1 Within a Bilingual SNMP Agent", RFC 2089, January 1997.

[10] Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1904, January 1996.

[11] McCloghrie, K. and F. Kastenholz, "Evolution of the Interfaces Group of MIB-II", RFC 1573, January 1994.

[12] Case, J., "FDDI Management Information Base", RFC 1285, January 1992.

[13] Application MIB Working Group, Krupczak, C., and J. Saperia, "Definitions of System-Level Managed Objects for Applications", draft-ietf-applmib-sysapplmib-08.txt, 15 Apr 1997.

## 13. Full Copyright Statement

Copyright (C) The Internet Society (1998). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS 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.