Network Working Group
Request for Comments: 3395
Updates: 2895
Category: Standards Track

A. Bierman
C. Bucci
Cisco Systems, Inc.
R. Dietz
Hifn, Inc.
A. Warth

September 2002

Remote Network Monitoring MIB Protocol Identifier Reference Extensions

Status of this Memo

Copyright Notice

Abstract

This memo defines extensions to the Protocol Identifier Reference document for the identification of application verb information. It updates the Protocol Identifier Reference document but does not obsolete any portion of that document. In particular, it describes the algorithms required to identify protocol operations (verbs) within the protocol encapsulations managed with MIBs such as the Remote Network Monitoring MIB Version 2, RFC 2021.

Table of Contents

1. The SNMP Network Management Framework

The SNMP Management Framework presently consists of five major components:

- An overall architecture, described in RFC 2571 [RFC2571].
- Mechanisms for describing and naming objects and events for the purpose of management. The first version of this Structure of Management Information (SMI) is called SMIv1 and is described in STD 16, RFC 1155 [RFC1155], STD 16, RFC 1212 [RFC1212] and

RFC 1215 [RFC1215]. The second version, called SMIv2, is described in STD 58, RFC 2578 [RFC2578], RFC 2579 [RFC2579] and RFC 2580 [RFC2580].

- Message protocols for transferring management information. The first version of the SNMP message protocol is called SNMPv1 and is described in STD 15, RFC 1157 [RFC1157]. A second version of the SNMP message protocol, which is not an Internet standards track protocol, is called SNMPv2c and is described in RFC 1901 [RFC1901] and RFC 1906 [RFC1906]. The third version of the message protocol is called SNMPv3 and is described in RFC 1906 [RFC1906], RFC 2572 [RFC2572] and RFC 2574 [RFC2574].

- Protocol operations for accessing management information. The first set of protocol operations and associated PDU formats is described in STD 15, RFC 1157 [RFC1157]. A second set of protocol operations and associated PDU formats is described in RFC 1905 [RFC1905].

- A set of fundamental applications is described in RFC 2573 [RFC2573]. The view-based access control mechanism is described in RFC 2575 [RFC2575].

A more detailed introduction to the current SNMP Management Framework can be found in RFC 2570 [RFC2570].

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

This memo does not specify a MIB module.

2. Overview

There is a need for a standardized way of identifying the protocol operations defined for particular application protocols. Different protocol operations can have very different performance characteristics, and it is desirable to collect certain metrics at this level of granularity. This memo defines extensions to the existing protocol identifier structure [RFC2895] and is intended to update, not obsolete, the existing protocol identifier encoding rules.

2.1 Protocol Identifier Framework

The RMON Protocol Identifier (PI) structure [RFC2895] allows for a variable number of layer identifiers. Each layer contributes 4 octets to the protocolDirID OCTET STRING and one octet to the

protocolDirParameters OCTET STRING. These two MIB objects comprise the index in the protocolDirTable [RFC2021] and represent a globally unique identifier for a particular protocol encapsulation (or set of encapsulations if the wild-card base layer is used).

2.2 Protocol Identifier Extensions for Application Verbs

The existing RMON protocol identifier architecture requires that an application verb be represented by one additional protocol layer, appended to the protocol identifier for the parent application. Since some application verbs are defined as strings which can exceed 4 octets in length, an integer mapping must be provided for each string. This memo specifies how the verb layer is structured, as well as a verb identifier macro syntax for specification of verb name to integer mappings.

2.3 Terms

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

This document uses some terms defined in the RMON Protocol Identifier Reference document [RFC2895] and some new terms that need introduction here.

Application Verb
: Also called simply 'verb'. Refers to one of potentially many protocol operations that are defined by a particular application protocol.

  Note that an application verb is not equivalent to an application protocol sub-command or opcode within a packet containing a PDU for the application. An application verb is a transaction type and may involve several PDU types within the application protocol (e.g., SNMP Get-PDU and Response-PDU). In some applications, a verb may encompass protocol operations pertaining to more than one protocol entry in the protocol directory (e.g., ftp and ftp-data).

Connect Verb
: The special application verb associated with connection or session setup and tear-down traffic, and not attributed to any other verb for the application. This verb is assigned the enumeration value of zero, and the verb 'connect(0)' is implicitly defined for all application protocols.

Parent Application
One of potentially many protocol encapsulations which identifies a particular application protocol. This term refers generically to any or all such encapsulations for a given set of application verbs.

Verb Layer
The portion of the protocol identifier octet string which identifies the application verb.

Verb Set
The group of verbs enumerated for a particular application protocol. The list of verb strings within a particular verb-identifier macro invocation is also called the verb set for that verb identifier.

2.4 Relationship to the RMON-2 MIB

The RMON-2 MIB [RFC2021] contains the protocolDirTable MIB objects used to identify all protocol encapsulations that can be monitored by a particular RMON agent.

This memo describes how these MIB objects are mapped by an implementation for entries which identify application verbs. This document does not define any new MIB objects to identify application verbs. The applicability of the definitions in this document is not limited to the RMON-2 MIB. Other specifications which utilize the RMON-2 protocolDirTable and/or the protocol identifier macros which it represents can also utilize the application verb macro definitions contained in this document.

2.5 Relationship to the RMON MIB Protocol Identifier Reference

The RMON MIB Protocol Identifier Reference [RFC2895] defines the RMON Protocol Identifier Macro Specification Language as well as the encoding rules for the ProtocolDirID and protocolDirParameters OCTET STRINGs. This memo defines extensions to the Protocol Identifier Reference for the identification of application verb information. It does not obsolete any portion of the Protocol Identifier Reference document.

3. Definitions

3.1 Verb Identifier Macro Format

The following example is meant to introduce the verb-identifier macro. This macro-like construct is used to represent protocol verbs for a specific parent application.

3.1.1 Lexical Conventions

The following keyword is added to the PI language:

```
VERB-IDENTIFIER
```

3.1.2 Extended Grammar for the PI Language

The following is the extended BNF notation for the grammar with starting symbol <piFile>. It is for representing verb identifier macros. Note that only the term <piFile> is actually modified from the definition in [RFC2895]. The <piDefinition> syntax is not reproduced here, since this memo is intended to extend that definition, not replace it.

```
-- a file containing one or more
-- Protocol Identifier (PI) definitions
<piFile> = [ <piDefinition> | <piVerbDefinition> ]...

-- a PI definition
<piVerbDefinition> =
  [<wspace>] <parentProtoName> <wspace> "VERB-IDENTIFIER"
        <wspace> "DESCRIPTION" <wspace> string
      [ <wspace> "REFERENCE" <wspace> string ]
      [<wspace>] "::=" [<wspace>]
      "{" [<wspace>] <verbList> [<wspace>] "}" [<wspace>]

-- a list of verb identifier string
<verbList> = <verbId> [ [<wspace>] "," [<wspace>] <verbId> ]...

-- a verb identifier string
<verbId> = <verbName> [<wspace>] "(" [<wspace>]
          <verbEnum> [<wspace>] ")" [<wspace>]

-- a protocol name
<parentProtoName> = <protoName>

-- a verb name
<verbName> = <lcname>

-- a verb enumeration
<verbEnum> = <posNum>

-- a positive integer
<posNum> = any integer value greater than zero and
           less than 16,777,216

-- <piDefinition> syntax is defined in [RFC2895]
```

```
-- <protoName> syntax is defined in [RFC2895]
-- <wspace> syntax is defined in [RFC2895]
-- <lcname> syntax is defined in [RFC2895]
```

3.1.3 Mapping of the Parent Protocol Name

The "parentProtoName" value, called the "parent protocol name", SHOULD be an ASCII string consisting of 1 to 64 characters. (These names are intended to appear in IETF documentation, so the use of UTF-8 is not appropriate.) The encoding rules are exactly as specified in section 6.2.4 of [RFC2895] for the mapping of the protocol name field. The value for <parentProtoName> (which is called the "parent protocol name") MUST be the value of a protocol identifier defined as specified for <protoName> in section 3.2.4 of [RFC2895]. The value of <parentProtoName> MUST specify a <protoName> defined in the <piFile>.

A protocol identifier macro SHOULD exist in the <piFile> for at least one encapsulation of the parent application protocol if any verb identifier macros referencing that parent application are present in the <piFile>.

3.1.4 Mapping of the DESCRIPTION Clause

The DESCRIPTION clause provides a textual description of the protocol verb set identified by this macro. It SHOULD NOT contain details about items covered by the REFERENCE clause. The DESCRIPTION clause MUST be present in all verb-identifier macro declarations.

3.1.5 Mapping of the REFERENCE Clause

If a publicly available reference document exists for this set of application protocol verbs, it SHOULD be listed here. Typically this will be a URL, otherwise it will be the name and address of the controlling body.

The REFERENCE clause is optional but SHOULD be present if an authoritative reference exists which specifies the application protocol verbs defined in the <verbList> section of this macro.

3.1.6 Mapping of the Verb List Clause

The verb list clause MUST be present. It is used to identify a list of application verb names and associate a numeric constant with each verb name. At least one verb MUST be specified and a maximum of 16,777,215 (2^^24 - 1) verbs MAY be specified. This enumerated list SHOULD be densely numbered (i.e., valued from '1' to 'N', where 'N' is the total number of verbs defined in the macro).

3.1.6.1 Mapping of the Verb Name Field

The <verbName> field is case-sensitive and SHOULD be set to the most appropriate string name for each application verb. If such a descriptive string is defined in an authoritative document then that string SHOULD be used. If no such string exists then an appropriate but arbitrary string should be selected for this value.

Verb names MUST be unique for a particular parent application. Note that the special 'connect(0)' verb is implicitly defined for each application protocol. It is possible for an explicit definition of this verb (e.g., 'connect(8)' for http) to exist for a protocol, as well as the implicit 'connect(0)' verb.

3.1.6.2 Mapping of the Verb Enum Field

The <verbEnum> field MUST be unique for all verbs associated with a particular parent application. This field SHOULD contain a value between '1' and '16,777,215' inclusive.

3.2 Protocol Directory Requirements

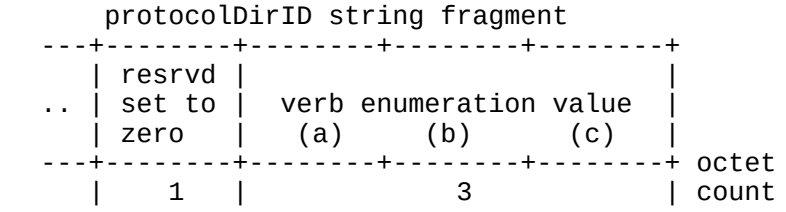
This section defines how the protocolDirTable should be populated for any application verb identified with a verb-identifier macro.

An agent MUST implement all applicable protocolDirTable MIB objects on behalf of each supported application verb.

3.2.1 Mapping of the Verb Layer Numbering Space

The verb layer consists of the 4 octets within the protocolDirID INDEX field which identify a particular application verb.

Figure 1
Verb Layer Format



The first octet is reserved for future use and MUST be set to zero.

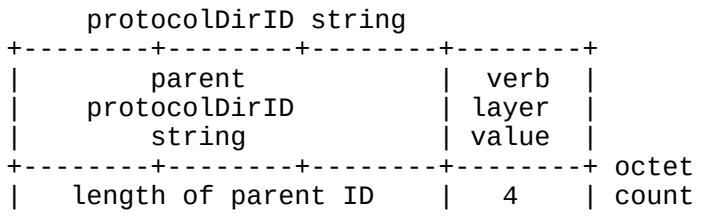
The next three octets identify the <verbEnum> field used to enumerate the particular application verb represented by the <verbName> field. This field is a 24-bit unsigned integer, encoded in network byte order.

The value zero is reserved to identify the special 'connect(0)' verb. This verb enumeration value (i.e., '0' part of 'connect(0)') MUST NOT be redefined in a verb identifier macro verb list. Note that the verb name 'connect' is not reserved and MAY be redefined in a verb list.

3.2.2 Mapping of the ProtocolDirID object

The protocolDirID OCTET STRING value for a particular application verb is represented by the protocolDirID value for the parent application, appended with the verb's layer identifier value.

Figure 2
ProtocolDirID Format for Verbs



The protocolDirID object is encoded as the protocolDirID value of the parent application, followed by four additional octets representing the verb layer. The verb layer value is encoded as [0.a.b.c] where 'a' is the high order byte, 'b' is the middle order byte, and 'c' is the low order byte of the <verbEnum> field for the specific application verb value. A valid PI verb enumeration will be encoded in the range "0.0.0.0" to "0.255.255.255", where the special value "0.0.0.0" is reserved for the implicitly defined 'connect(0)' verb.

3.2.3 Mapping of the ProtocolDirParameters object

The protocolDirParameters OCTET STRING value for a particular application verb is represented by the protocolDirParameters value for the parent application, appended with one octet containing the value zero. Although not actually used, this field is included to conform to the encoding rules defined in the Protocol Identifiers Reference [RFC2895].

3.2.4 Mapping of the ProtocolDirLocalIndex object

The agent MUST assign an appropriate protocolDirLocalIndex value for each application verb according to the encoding rules defined for this object in [RFC2021] and [RFC2895].

3.2.5 Mapping of the protocolDirDescr object

The agent MUST convey the <verbName> value for a particular application verb in the protocolDirDescr object. This object SHOULD be encoded as the protocolDirDescr value for the parent application appended with a 'dot' character, followed by the exact text contained in the <verbName> field.

3.2.6 Mapping of the protocolDirType object

The agent MUST set the protocolDirType object for each application verb to the value representing the empty bit set ({}).

3.2.7 Mapping of the protocolDirAddressMapConfig object

The agent MUST set the protocolDirAddressMapConfig object for each application verb to the value 'notSupported(1)'.

3.2.8 Mapping of the protocolDirHostConfig object

The agent MUST set the protocolDirHostConfig object for each application verb present in the protocol directory according to the monitoring capabilities for each verb. The agent MAY set this object to the same value as configured in the parent application protocolDirHostConfig object. The agent MAY choose to transition this object from the value 'supportedOn(2)' to 'supportedOff(3)' if the parent application protocolDirHostConfig object first transitions from 'supportedOn(2)' to 'supportedOff(3)'.

3.2.9 Mapping of the protocolDirMatrixConfig object

The agent MUST set the protocolDirMatrixConfig object for each application verb according to the monitoring capabilities for each verb. The agent MAY set this object to the same value as configured in the parent application protocolDirMatrixConfig object. The agent MAY choose to transition this object from the value 'supportedOn(2)' to 'supportedOff(3)' if the parent application protocolDirMatrixConfig object first transitions from 'supportedOn(2)' to 'supportedOff(3)'.

3.2.10 Mapping of the protocolDirOwner object

This object is encoded exactly the same for application verbs as for other protocolDirTable entries, according to the rules specified in the RMON-2 MIB [RFC2021].

3.2.11 Mapping of the protocolDirStatus object

This object is encoded exactly the same for application verbs as for other protocolDirTable entries, according to the rules specified in RMON-2 MIB [RFC2021].

4. Implementation Considerations

This section discusses the implementation implications for agents which support verbs in the protocol directory and the RMON collections which utilize the protocol directory.

4.1 Stateful Protocol Decoding

Implementations of the RMON-2 MIB for application layer and network layer protocols typically require little if any state to be maintained by the probe. The probe can generally decide whether to count a packet and its octets on the packet's own merits, without referencing or updating any state information.

Implementations of the RMON-2 MIB at the verb layer will, for many protocols, need to maintain state information in order to correctly classify a packet as "belonging" to one verb or another. The examples below illustrate this point.

For SNMP over UDP, a Response-PDU for an SNMP Get-PDU can't be distinguished from a Response-PDU for a Getnext-PDU. A probe would need to maintain state information in order to correlate a Response-PDU from B to A with a previous request from A to B.

For application protocols carried over a stream-based transport such as TCP, the information required to identify an application verb can span several packets. A probe would need to follow the transport-layer flow in order to correctly parse the application-layer data.

4.2 Packet Capture

For packet capture based on verb-layer protocol directory filtering, the decision to include a packet in the capture buffer may need to be deferred until the packet can be conclusively attributed to a

particular verb. A probe may need to pre-buffer packets while deciding to include or exclude them from capture based on other packets that have not yet arrived.

4.3 RMON-2 MIB Collections

Data collections such as the protocol distribution or Application Layer Host Table (alHostTable) require that each packet is counted only once, i.e., a given packet is fully classified as a single protocol encapsulation which resolves to a single leaf entry in the protocol directory. Also, octet counters related to protocol classification are incremented by the entire size of packet, not just the octets associated with a particular encapsulation layer.

It is possible that particular application protocols will allow multiple types of verbs to be present in a single packet. In this case, the agent MUST choose one verb type, and therefore one protocol directory entry, in order to properly count such a packet.

It is an implementation-specific matter as to which verb type an agent selects to identify a packet in the event more than one verb type is present in that packet. Some possible choices include:

- the first verb type encountered in the packet
- the verb type with the most instances in the packet
- the verb type using the largest number of octets in the packet
- the most 'interesting' verb type in the packet (based on knowledge of that application protocol).

5. Intellectual Property

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

6. Acknowledgements

This memo is a product of the RMONMIB WG.

7. Normative References

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

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

[RFC2021] Waldbusser, S., "Remote Network Monitoring MIB (RMON-2)", RFC 2021, January 1997.

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

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

[RFC2571] Harrington, D., Presuhn, R. and B. Wijnen, "An Architecture for Describing SNMP Management Frameworks", RFC 2571, April 1999.

[RFC2572] Case, J., Harrington D., Presuhn R. and B. Wijnen, "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)", RFC 2572, April 1999.

[RFC2573] Levi, D., Meyer, P. and B. Stewart, "SNMPv3 Applications", RFC 2573, April 1999.

[RFC2574] Blumenthal, U. and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", RFC 2574, April 1999.

[RFC2575] Wijnen, B., Presuhn, R. and K. McCloghrie, "View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)", RFC 2575, April 1999.

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

[RFC2579] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.

[RFC2580] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.

[RFC2895] Bierman, A., Bucci, C. and R. Iddon, "Remote Network Monitoring MIB Protocol Identifiers", RFC 2895, August 2000.

8. Informative References

[RFC1155] Rose, M. and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based Internets", STD 16, RFC 1155, May 1990.

[RFC1157] Case, J., Fedor, M., Schoffstall, M. and J. Davin, "Simple Network Management Protocol", STD 15, RFC 1157, May 1990.

[RFC1212] Rose, M. and K. McCloghrie, "Concise MIB Definitions", STD 16, RFC 1212, March 1991.

[RFC1215] Rose, M., "A Convention for Defining Traps for use with the SNMP", RFC 1215, March 1991.

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

[RFC2570] Case, J., Mundy, R., Partain, D. and B. Stewart, "Introduction to Version 3 of the Internet-standard Network Management Framework", RFC 2570, April 1999.

9. IANA Considerations

At this time there are no application protocol verbs defined that require IANA registration, similar to the 'ianaAssigned' protocol identifiers found in RFC 2895. It is remotely possible that a future version of this document will contain application verb definitions which require assignment in the 'ianaAssigned' protocol identifier subtree.

10. Security Considerations

This memo defines the structure of a portion of the Remote Monitoring MIB framework, but does not define any MIB objects or protocol operations. Instead, it defines algorithms for representing application protocol verbs in RMON Protocol Identifiers. It does not introduce any new security risks into a managed system.

However, if an MIB collection is designed which utilizes this type of Protocol Identifier, then such a collection may expose which verbs in an application protocol are used in a network. Inclusion of this additional information may require more consideration for protection. MIB writers should address such considerations.

Appendix A: Usage Examples

The following examples are listed to demonstrate how RMON verb identifiers are declared.

A.1 FTP Example

This example defines verb enumeration values for the File Transfer Protocol as defined in RFC 959 and updated by RFC 2228 and RFC 2640. Note that verb name strings specified in the <verbName> field are not limited to 4 characters in length. In the FTP protocol, all the command names are 4 characters in length and the verb name string should match the official command name as closely as possible.

```
ftp VERB-IDENTIFIER
    DESCRIPTION
      "The set of verbs for FTP is derived from the list
       of commands defined for the File Transfer Protocol,
       which are identified by case-insensitive strings.
       The commands are simply listed in the order found
       in the FTP documentation."
    REFERENCE
      "File Transfer Protocol, RFC 959, Section 4.1;
       FTP Security Extensions, RFC 2228, Section 3;
       Internationalization of the File Transfer Protocol,
       RFC 2640, Section 4.1."
    ::= {
          user(1),     -- USER NAME
          pass(2),     -- PASSWORD
          acct(3),     -- ACCOUNT
          cwd(4),      -- CHANGE WORKING DIRECTORY
          cdup(5),     -- CHANGE TO PARENT DIRECTORY
          smnt(6),     -- STRUCTURE MOUNT
          rein(7),     -- REINITIALIZE
          quit(8),     -- LOGOUT
          port(9),     -- DATA PORT
          pasv(10),    -- PASSIVE
          type(11),    -- REPRESENTATION TYPE
          stru(12),    -- FILE STRUCTURE
          mode(13),    -- TRANSFER MODE
          retr(14),    -- RETRIEVE
          stor(15),    -- STORE
          stou(16),    -- STORE UNIQUE
          appe(17),    -- APPEND (with create)
          allo(18),    -- ALLOCATE
          rest(19),    -- RESTART
          rnfr(20),    -- RENAME FROM
          rnto(21),    -- RENAME TO
```

```
            abor(22),    -- ABORT
            dele(23),    -- DELETE
            rmd(24),     -- REMOVE DIRECTORY
            mkd(25),     -- MAKE DIRECTORY
            pwd(26),     -- PRINT WORKING DIRECTORY
            list(27),    -- LIST
            nlst(28),    -- NAME LIST
            site(29),    -- SITE PARAMETERS
            syst(30),    -- SYSTEM
            stat(31),    -- STATUS
            help(32),    -- HELP
            noop(33),    -- NOOP
            auth(34),    -- AUTHENTICATION/SECURITY MECHANISM
            adat(35),    -- AUTHENTICATION/SECURITY DATA
            pbsz(36),    -- PROTECTION BUFFER SIZE
            prot(37),    -- DATA CHANNEL PROTECTION LEVEL
            ccc(38),     -- CLEAR COMMAND CHANNEL
            mic(39),     -- INTEGRITY PROTECTED COMMAND
            conf(40),    -- CONFIDENTIALITY PROTECTED COMMAND
            enc(41),     -- PRIVACY PROTECTED COMMAND
            lang(42)     -- LANGUAGE
    }
```

A.2 POP3 Example

This example defines verb enumeration values for the Post Office Protocol, Version 3, as defined in RFC 1939 and updated by RFC 2449.

```
pop3 VERB-IDENTIFIER
    DESCRIPTION
      "The set of verbs for POP3 is derived from the list
       of commands defined for the Post Office Protocol,
       which are identified by case-insensitive strings.
       The commands are simply listed in the order found
       in the POP3 command summary."
    REFERENCE
      "Post Office Protocol, Version 3, RFC 1939, Section 9;
       POP3 Extension Mechanism, RFC 2449, Section 5."
    ::= {
          user(1),
          pass(2),
          quit(3),
          stat(4),
          list(5),
          retr(6),
          dele(7),
          noop(8),
          rset(9),
```

```
            apop(10),
            top(11),
            uidl(12),
            capa(13)
        }
```

A.3 SNMP Example

This example defines verb enumeration values for the Simple Network Management Protocol, as defined in RFC 1905.

```
snmp VERB-IDENTIFIER
    DESCRIPTION
      "The set of verbs for SNMP is derived from the list
       of PDU transaction types in the Protocol Operations
       document for SNMPv2.  Note that the 'Response'
       and 'Report' PDUs are not considered verbs, but are
       classified as belonging to the transaction type
       associated with the request PDU."
    REFERENCE
      "Protocol Operations for Version 2 of the
       Simple Network Management Protocol (SNMPv2),
       RFC 1905, Section 3."
    ::= {
          get(1),
          get-next(2),
          get-bulk(3),
          set(4),
          inform-request(5),
          trap(6)
    }
```

A.4 HTTP Example

This example defines verb enumeration values for the Hypertext Transfer Protocol, version 1.1, as defined in RFC 2616.

```
http VERB-IDENTIFIER
    DESCRIPTION
      "The set of verbs for HTTP is derived from the list
       of methods defined for the Hypertext Transfer Protocol,
       which are identified by case-sensitive strings.
       The commands are simply listed in the order found
       in the HTTP/1.1 documentation.  Methods commonly used
       in HTTP/1.0 are a proper subset of those used in HTTP/1.1.
       Both versions of the protocol are in current use."
    REFERENCE
       "Hypertext Transfer Protocol -- HTTP/1.1, RFC 2616,
```

```
          Section 9; Hypertext Transfer Protocol -- HTTP/1.0, RFC
          1945, Section 8."
       ::= {
             options(1),
             get(2),
             head(3),
             post(4),
             put(5),
             delete(6),
             trace(7),
             connect(8)  -- reserved for future use by HTTP/1.1
       }
```

A.5 SMTP Example

This example defines verb enumeration values for the Simple Mail Transfer Protocol as defined in RFC 2821.

```
smtp VERB-IDENTIFIER
    DESCRIPTION
    "The set of verbs for SMTP is derived from the set of commands
     defined for the protocol.  These commands are identified
     by case-insensitive strings.  Commands are listed in the
     order found in RFC 2821.  The special "xcmd" verb is defined
     here as a catch-all for private-use commands, which must
     start with the letter 'X'."
    REFERENCE
      "Simple Mail Transfer Protocol -- RFC 2821, sections 4.1.1
       and 4.1.5."
    ::= {
          ehlo(1),  -- Extended HELLO (4.1.1.1)
          helo(2),  -- HELLO (4.1.1.1)
          mail(3),  -- MAIL (4.1.1.2)
          rcpt(4),  -- RECIPIENT (4.1.1.3)
          data(5),  -- DATA (4.1.1.4)
          rset(6),  -- RESET (4.1.1.5)
          vrfy(7),  -- VERIFY (4.1.1.6)
          expn(8),  -- EXPAND (4.1.1.7)
          help(9),  -- HELP (4.1.1.8)
          noop(10), -- NOOP (4.1.1.9)
          quit(11), -- QUIT (4.1.1.10)
          xcmd(12)  -- Catch-all for private-use "X" commands (4.1.5)
    }
```

Authors' Addresses

Andy Bierman
Cisco Systems, Inc.
170 West Tasman Dr
San Jose, CA USA 95134

Phone: +1 408-527-3711
EMail: abierman@cisco.com

Chris Bucci
Cisco Systems, Inc.
170 West Tasman Dr
San Jose, CA USA 95134

Phone: +1 408-527-5337
EMail: cbucci@cisco.com

Russell Dietz
Hifn, Inc.
750 University Ave
Los Gatos, CA, USA 95032-7695

Phone: +1 408-399-3623
EMail: rdietz@hifn.com

Albin Warth

EMail: dahoss@earthlink.net

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.