Internet Endpoint MIB

<draft-ops-endpoint-mib-01.txt>

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

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

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

Abstract

This MIB module defines constructs to represent commonly used addressing information. The intent is that these definitions will be imported and used in the various MIBs that would otherwise define their own representations. This work is output from the Operations and Management Area "IPv6MIB" design team.

1. The SNMP Management Framework

The SNMP Management Framework presently consists of five major components:

- o   An overall architecture, described in RFC 2571 [RFC2571].

- o   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 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].

- o Message protocols for transferring management information. The first version of the SNMP message protocol is called SNMPv1 and 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 described in RFC 1901 [RFC1901] and RFC 1906 [RFC1906]. The third version of the message protocol is called SNMPv3 and described in RFC 1906 [RFC1906], RFC 2572 [RFC2572] and RFC 2574 [RFC2574].

- o 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].

- o A set of fundamental applications described in RFC 2573 [RFC2573] and the view-based access control mechanism 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 specifies a MIB module that is compliant to the SMIv2. A MIB conforming to the SMIv1 can be produced through the appropriate translations. The resulting translated MIB must be semantically equivalent, except where objects or events are omitted because no translation is possible (use of Counter64). Some machine readable information in SMIv2 will be converted into textual descriptions in SMIv1 during the translation process. However, this loss of machine readable information is not considered to change the semantics of the MIB.

2. Definitions

```
INET-ENDPOINT-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY FROM SNMPv2-SMI
    TEXTUAL-CONVENTION FROM SNMPv2-TC;

inetEndpointMIB MODULE-IDENTITY
   LAST-UPDATED "9910210000Z"
   ORGANIZATION "IETF OPS Area"
   CONTACT-INFO "Send comments to mibs@ops.ietf.org"
   DESCRIPTION
```

```
        "A MIB module for Internet address definitions."
   ::= { ??? }


--
--
-- New TCs for representing generic Internet endpoints.
-- These are roughly equivalent to TDomain and TAddress...
--
--

--
-- Internet endpoints types
--
InetEndpointType ::= TEXTUAL-CONVENTION
    STATUS       current
    DESCRIPTION
          "A value that represents a type of Internet endpoint.

           Note that it is possible to sub-type objects defined with
           this syntax by removing one or more enumerated values.
           The DESCRIPTION clause of such objects (or their corresponding
           InetEndpoint object) must document specific usage."
    SYNTAX      INTEGER {
                        other(0),
                        ipv4(1),
                        ipv6(2),
                        dns(3)
                        }

InetEndpoint ::= TEXTUAL-CONVENTION
    STATUS       current
    DESCRIPTION
          "Denotes an generic Internet endpoint.

          A InetEndpoint value is always interpreted within the context of a
          InetEndpointType value.  Thus, each definition of a InetEndpointType
          value must be accompanied by a definition of a textual convention
          for use with that InetEndpointType.

          When this Textual Convention is used as the syntax of an index
object,
          there may be issues with the limit of 128 sub-identifiers specified
          in [SMIv2].  In this case, it is recommended that the OBJECT-TYPE
          declaration include a 'SIZE' clause to limit the number of potential
          instance sub-identifiers."
    REFERENCE "See the TAddress TC in std58."
    SYNTAX       OCTET STRING (SIZE (0..255))


--
```

```
--
-- TCs for specific Internet endpoint values.
--
--

--
-- IPv4 Address
--

InetEndpointIPv4 ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "1d.1d.1d.1d"
    STATUS       current
    DESCRIPTION
            "Represents an IPv4 network address:

               octets   contents        encoding
                1-4     IP address      network-byte order

             The corresponding InetEndpointType is ipv4(1)."
    SYNTAX       OCTET STRING (SIZE (4))

--
-- IPv6 Address
--

InetEndpointIPv6 ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "2x:2x:2x:2x:2x:2x:2x:2x"
    STATUS       current
    DESCRIPTION
            "Represents an IPv6 network address:

                octets   contents        encoding
                 1-16    IPv6 address    network-byte order

             The corresponding InetEndpointType is ipv6(2)."
     REFERENCE "See the Ipv6Address TC in RFC 2465."
     SYNTAX       OCTET STRING (SIZE (16))

--
-- DNS Name
--

InetEndpointDNS ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "255a"
    STATUS       current
    DESCRIPTION
            "Represents a fully qualified DNS host name.
             The corresponding InetEndpointType is dns(3).

             The DESCRIPTION clause of InetEndpoint objects that
             may have InetEndpointDNS values must fully describe
             how (and when) such names are to be resolved to IP
```

```
        addresses."
    REFERENCE "RFCs 952 and 1123."
    SYNTAX       OCTET STRING (SIZE (1..255))

END
```

3. Usage

These definitions provide a mechanism to define generic Internet-accessible endpoints within MIB specifications. It is recommended that MIB developers use these definitions when applicable, as opposed to defining their own constructs.

A generic Internet endpoint consists of two objects, one whose syntax is InetEndpointType, and another whose syntax is InetEndpoint. The value of the first object determines how the value of the second object is encoded.

One particular usage of InetEndpointType/InetEndpoint pairs is to avoid over-constraining an object definition by the use of the IpAddress syntax. IpAddress limits an implementation to using IPv4 addresses only, and as such SHOULD only be used when the object truly is IPv4-specific.

4. Indexing

When a generic Internet endpoint is used as an index, both the InetEndpointType and InetEndpoint objects MUST be used, and the InetEndpointType object MUST come first in the INDEX clause.

The InetEndpointType object may be subtyped such that the resulting index is of fixed length. But the more common usage will result in variable-length indexes.

For variable length indexes, the IMPLIED keyword MUST NOT be used in the INDEX clause. Instance subidentifiers are then of the form T.N.O1.O2...On, where T is the value of the InetEndpointType object, O1...On are the octets in the InetEndpoint object, and N is the number of those octets.

There is a meaningful lexicographical ordering to tables indexed in this fashion. Command generator applications may

- o lookup specific endpoints of known type and value
- o issue GetNext requests for endpoints of a single type
- o issue GetNext requests for specific type and address prefix

It should be pointed out that another valid approach is to define separate tables for different address types. For example, one table might be indexed by an IpAddress object, and the other table indexed by an Ipv6Address object. This is a decision for the MIB designer. (For example, the tcpConnTable was left intact and a new table added for TCP connections over IPv6, see RFC 2452.)

5. Uniqueness of Addresses

IPv4 addresses were intended to be globally unique, current usage notwithstanding. IPv6 addresses were architected to have different scopes and hence uniqueness. In particular, IPv6 "link-local" and "site-local" addresses are not guaranteed to be unique on any particular node. In such cases, the duplicate addresses must be configured on different interfaces, so the combination of IPv6 address/interface is unique.

For tables indexed by InetEndpointType/InetEndpoint pairs, where there may be non-unique instances of InetEndpointIPv6, the recommended approach is to add a third index object to ensure uniqueness.

It is recommended that the syntax of this third index object be InterfaceIndexOrZero, imported from IF-MIB [RFC2233]. The value of this object SHOULD be 0 when the value of the InetEndpointType object is not ipv6(2).

6. Multiple InetEndpoints per Host

Note that a single host system may be configured with multiple addresses (IPv4 or IPv6), and possibly with multiple DNS names. Thus it is possible for a single host system to be represented by multiple (unique) InetEndpointType/InetEndpoint pairs.

If this could be an implementation or usage issue the DESCRIPTION clause of the relevant objects MUST fully describe required behavior.

7. Resolving DNS Names

DNS names are translated to IP addresses when communication with a host is required. This raises a temporal aspect to defining MIB objects whose value is a DNS name; when is the name translated to an address?

For example, consider an object defined to indicate a forwarding destination, and whose value is a DNS name. When does the forwarding entity resolve the DNS name? Each time forwarding occurs? Once, when the object was instantiated?

The DESCRIPTION clause of such objects SHOULD precisely define how (when) any required name to address resolution is done.

8. Usage Examples

Example 1:

```
    fooTable OBJECT-TYPE
```

```
            SYNTAX      SEQUENCE OF FooEntry
            MAX-ACCESS  not-accessible
            STATUS      current
            DESCRIPTION
            "The foo table."
            ::= { bar 1 }

    fooEntry OBJECT-TYPE
            SYNTAX      FooEntry
            MAX-ACCESS  not-accessible
            STATUS      current
            DESCRIPTION
            "A foo entry."
            INDEX      { fooPartnerType, fooPartner }
            ::= { fooTable 1 }

    FooEntry ::= SEQUENCE {
            fooPartnerType  InetEndpointType,
            fooPartner      InetEndpoint,
            fooStatus       INTEGER,
            fooDescr        OCTET STRING
    }

    fooPartnerType ::= OBJECT-TYPE
            SYNTAX      InetEndpointType
            MAX-ACCESS  not-accessible
            STATUS      current
            DESCRIPTION
            "The type of Internet endpoint by which the partner is
reachable."
            ::= { fooEntry 1 }

    fooPartner ::= OBJECT-TYPE
            SYNTAX      InetEndpoint (SIZE (0..64))
            MAX-ACCESS  not-accessible
            STATUS      current
            DESCRIPTION
            "The Internet endpoint for the partner.  Note that
implementations
             must limit themselves to a single entry in this table per
reachable
             partner.  Also, if an Ipv6 endpoint is used, it must contain a
globally
             unique IPv6 address."
            ::= { fooEntry 2 }
```

Example 2:

```
    sysAddrTable OBJECT-TYPE
            SYNTAX      SEQUENCE OF SysAddrEntry
            MAX-ACCESS  not-accessible
```

```
        STATUS      current
        DESCRIPTION
        "The sysAddr table."
        ::= { sysAddr 1 }

sysAddrEntry OBJECT-TYPE
        SYNTAX      SysAddrEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
        "A sysAddr entry."
        INDEX      { sysAddrType, sysAddr, sysAddrIfIndex }
        ::= { sysAddrTable 1 }

SysAddrEntry ::= SEQUENCE {
        sysAddrPartnerType      InetEndpointType,
        sysAddrPartner          InetEndpoint,
        sysAddrIfIndex          InterfaceIndexOrZero,
        sysAddrStatus           INTEGER,
        sysAddrDescr            OCTET STRING
}

sysAddrType ::= OBJECT-TYPE
        SYNTAX      InetEndpointType {
                          ipv4(1),
                          ipv6(2)
                    }
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
        "The type of system address."
        ::= { sysAddrEntry 1 }

sysAddr ::= OBJECT-TYPE
        SYNTAX      InetEndpoint (SIZE (4 | 16))
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
        "The system address."
        ::= { sysAddrEntry 2 }

sysAddrIfIndex ::= OBJECT-TYPE
        SYNTAX     InterfaceIndexOrZero
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
        "The system address interface.  This object is used to
disambiguate
         duplicate system IPv6 addresses, and should be 0 for non-
duplicate
         addresses."
```

```
            ::= { sysAddrEntry 3 }
```

9. References

[RFC2233] K. McCloghrie, and F. Kastenholz, "The Interfaces Group MIB using SMIv2", RFC 2233, November 1997

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

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

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

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

[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

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

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

[RFC1906] 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.

[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

[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

[RFC1905] 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.

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

[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

10. Authors

This work was done by the IETF Ops Area "IPv6MIB" Design Team. Comments should be posted to mibs@ops.ietf.org.

11. Notices

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.

12. Full Copyright Statement

13. Appendix A

This appendix lists the issues raised over common addressing MIB constructs, and the reasoning for the decisions made in this module.

1. Efficient table lookups

   Some existing MIBs have tables of generic addresses, indexed by a random integer. This makes it impossible to lookup specific addresses, or issue meaningful GetNext operations.

2. Common addressing should be defined such that no SMI changes are required.

   For example, the use of the ASN.1 CHOICE would really be an SMI change.

3. TCs and DISPLAY-HINTS

   A single object that contains both address type and value does not provide a way to express the display characteristics of each type.

   (Also, such a single object requires code changes to handle updates, whereas the solution chosen requires only MIB updates.)

4. Document the possible non-uniqueness of IPv6 addresses, and the impact on indexing tables.

5. TDomain/TAddress limited to transport services

   It was unclear if network layer addresses were appropriate for use in TAddress values, since std58 refers specifically to "transport addresses".

This point is less important than std58's definition that TAddress values always be defined in the context of TDomain values. Since did not want to index by OIDs, we did not use TDomain and hence cannot use TAddress.

6. Harness the use of IpAddress

   Several standard-track MIBs have used IpAddress syntax inadvertently, needlessly limiting implementations to IPv4.

   The specification under development should address this.

7. DNS names in addition to addresses

   It is useful to be able to specify a system via a DNS name, so the common addressing mechanism should support them.