

Internet Endpoint MIB

<[draft-ops-endpoint-mib-00.txt](#)>

1. 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/lid-abstracts.txt>

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

2. 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.

3. Definitions

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

```
IMPORTS
```

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

```
inetEndpointMIB MODULE-IDENTITY
```

```
    LAST-UPDATED "9907300000Z"
```

```
    ORGANIZATION "IETF OPS Area"
```

```
    CONTACT-INFO "Send comments to mibs@ops.ietf.org"
```

```
    DESCRIPTION
```

```

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

--
--
-- 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

[4. 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.

[5. 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.

Instance subidentifiers are then of the form `T.N.01.02...0n`, where `T` is the value of the `InetEndpointType` object, `01...0n` 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](#).)

[6. 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, from IF-MIB. The value of this object should be 0 when the value of the InetEndpointType object is not ipv6(2).

<< TBD: what about Ipv6IfIndexOrZero in [RFC 2465](#)? >>

7. 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 should fully describe required behavior.

8. 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.

9. 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
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 }
```

[10.](#) References

TBD

[11.](#) Copyright

TBD

12. Authors

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

Appendix

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.

Expires February, 2000