

Network Working Group  
Request for Comments: 3291  
Obsoletes: 2851  
Category: Standards Track

M. Daniele  
Consultant  
B. Haberman  
Consultant  
S. Routhier  
Wind River Systems, Inc.  
J. Schoenwaelder  
TU Braunschweig  
May 2002

## Textual Conventions for Internet Network Addresses

### 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 (2002). All Rights Reserved.

### Abstract

This MIB module defines textual conventions to represent commonly used Internet network layer addressing information. The intent is that these textual conventions (TCs) will be imported and used in MIB modules that would otherwise define their own representations.

This document obsoletes RFC 2851.

## Table of Contents

1. Introduction . . . . .	2
2. The SNMP Management Framework . . . . .	4
3. Definitions . . . . .	5
4. Usage Hints . . . . .	11
4.1 Table Indexing . . . . .	12
4.2 Uniqueness of Addresses . . . . .	12
4.3 Multiple Addresses per Host . . . . .	13
4.4 Resolving DNS Names . . . . .	13
5. Table Indexing Example . . . . .	13
6. Security Considerations . . . . .	16
7. Acknowledgments . . . . .	16
8. Intellectual Property Notice . . . . .	16
9. Changes from RFC 2851 . . . . .	16
References . . . . .	17
Authors' Addresses . . . . .	19
Full Copyright Statement . . . . .	20

## 1. Introduction

Several standards-track MIB modules use the `IpAddress` SMIV2 base type. This limits the applicability of these MIB modules to IP Version 4 (IPv4) since the `IpAddress` SMIV2 base type can only contain 4 byte IPv4 addresses. The `IpAddress` SMIV2 base type has become problematic with the introduction of IP Version 6 (IPv6) addresses [19].

This document defines multiple textual conventions as a mechanism to express generic Internet network layer addresses within MIB module specifications. The solution is compatible with SMIV2 (STD 58) and SMIV1 (STD 16). New MIB definitions which need to express network layer Internet addresses SHOULD use the textual conventions defined in this memo. New MIB modules SHOULD NOT use the SMIV2 `IpAddress` base type anymore.

A generic Internet address consists of two objects, one whose syntax is `InetAddressType`, and another whose syntax is `InetAddress`. The value of the first object determines how the value of the second object is encoded. The `InetAddress` textual convention represents an opaque Internet address value. The `InetAddressType` enumeration is used to "cast" the `InetAddress` value into a concrete textual convention for the address type. This usage of multiple textual conventions allows expression of the display characteristics of each address type and makes the set of defined Internet address types extensible.

The textual conventions defined in this document can also be used to represent generic Internet subnets and Internet address ranges. A generic Internet subnet is represented by three objects, one whose syntax is `InetAddressType`, a second one whose syntax is `InetAddress` and a third one whose syntax is `InetAddressPrefixLength`. The `InetAddressType` value again determines the concrete format of the `InetAddress` value while the `InetAddressPrefixLength` identifies the Internet network address prefix.

A generic range of consecutive Internet addresses is represented by three objects. The first one has the syntax `InetAddressType` while the remaining objects have the syntax `InetAddress` and specify the start and end of the address range. The `InetAddressType` value again determines the format of the `InetAddress` values.

The textual conventions defined in this document can be used to define Internet addresses by using DNS domain names in addition to IPv4 and IPv6 addresses. A MIB designer can write compliance statements to express that only a subset of the possible address types must be supported by a compliant implementation.

MIB developers who need to represent Internet addresses SHOULD use these definitions whenever applicable, as opposed to defining their own constructs. Even MIB modules that only need to represent IPv4 or IPv6 addresses SHOULD use the `InetAddressType/InetAddress` textual conventions defined in this memo.

There are many widely deployed MIB modules that use IPv4 addresses and which need to be revised to support IPv6. These MIBs can be categorized as follows:

1. MIB modules which define management information that is in principle IP version neutral, but the MIB currently uses addressing constructs specific to a certain IP version.
2. MIB modules which define management information that is specific to particular IP version (either IPv4 or IPv6) and which is very unlikely to ever be applicable to another IP version.

MIB modules of the first type SHOULD provide object definitions (e.g., tables) that work with all versions of IP. In particular, when revising a MIB module which contains IPv4 specific tables, it is suggested to define new tables using the textual conventions defined in this memo which support all versions of IP. The status of the new tables SHOULD be "current" while the status of the old IP version specific tables SHOULD be changed to "deprecated". The other approach of having multiple similar tables for different IP versions is strongly discouraged.

MIB modules of the second type, which are inherently IP version specific, do not need to be redefined. Note that even in this case, any additions to these MIB modules or new IP version specific MIB modules SHOULD use the textual conventions defined in this memo.

MIB developers SHOULD NOT use the textual conventions defined in this document to represent generic transport layer addresses. Instead the SMIV2 TAddress textual convention and associated definitions should be used for transport layer addresses.

The key words "MUST", "MUST NOT", "SHOULD", "SHOULD NOT" and "MAY" in this document are to be interpreted as described in RFC 2119 [1].

## 2. The SNMP Management Framework

The SNMP Management Framework presently consists of five major components:

- o An overall architecture, described in RFC 2571 [2].
- 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 [3], STD 16, RFC 1212 [4] and RFC 1215 [5]. The second version, called SMIV2, is described in STD 58, RFC 2578 [6], STD 58, RFC 2579 [7] and STD 58, RFC 2580 [8].
- 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 [9]. A second version of the SNMP message protocol, which is not an Internet standards track protocol, is called SNMPv2c and described in RFC 1901 [10] and RFC 1906 [11]. The third version of the message protocol is called SNMPv3 and described in RFC 1906 [11], RFC 2572 [12] and RFC 2574 [13].
- o Protocol operations for accessing management information. The first set of protocol operations and associated PDU formats is described in STD 15, RFC 1157 [9]. A second set of protocol operations and associated PDU formats is described in RFC 1905 [14].
- o A set of fundamental applications described in RFC 2573 [15] and the view-based access control mechanism described in RFC 2575 [16].

A more detailed introduction to the current SNMP Management Framework can be found in RFC 2570 [17].

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.

### 3. Definitions

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

```
IMPORTS
```

```
    MODULE-IDENTITY, mib-2, Unsigned32 FROM SNMPv2-SMI
    TEXTUAL-CONVENTION                                FROM SNMPv2-TC;
```

```
inetAddressMIB MODULE-IDENTITY
```

```
    LAST-UPDATED "200205090000Z"
```

```
    ORGANIZATION
```

```
        "IETF Operations and Management Area"
```

```
    CONTACT-INFO
```

```
        "Juergen Schoenwaelder (Editor)
```

```
        TU Braunschweig
```

```
        Bueltenweg 74/75
```

```
        38106 Braunschweig, Germany
```

```
        Phone: +49 531 391-3289
```

```
        EMail: schoenw@ibr.cs.tu-bs.de
```

```
        Send comments to <mibs@ops.ietf.org>."
```

```
DESCRIPTION
```

```
"This MIB module defines textual conventions for
representing Internet addresses. An Internet
address can be an IPv4 address, an IPv6 address
or a DNS domain name. This module also defines
textual conventions for Internet port numbers,
autonomous system numbers and the length of an
Internet address prefix."
```

```
REVISION      "200205090000Z"
```

```
DESCRIPTION
```

```
"Second version, published as RFC 3291. This
revisions contains several clarifications and it
```

introduces several new textual conventions:  
InetAddressPrefixLength, InetPortNumber,  
InetAutonomousSystemNumber, InetAddressIPv4z,  
and InetAddressIPv6z."

REVISION "200006080000Z"

DESCRIPTION

"Initial version, published as RFC 2851."

::= { mib-2 76 }

InetAddressType ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"A value that represents a type of Internet address.

unknown(0) An unknown address type. This value MUST  
be used if the value of the corresponding  
InetAddress object is a zero-length string.  
It may also be used to indicate an IP address  
which is not in one of the formats defined  
below.

ipv4(1) An IPv4 address as defined by the  
InetAddressIPv4 textual convention.

ipv6(2) A global IPv6 address as defined by the  
InetAddressIPv6 textual convention.

ipv4z(3) A non-global IPv4 address including a zone  
index as defined by the InetAddressIPv4z  
textual convention.

ipv6z(4) A non-global IPv6 address including a zone  
index as defined by the InetAddressIPv6z  
textual convention.

dns(16) A DNS domain name as defined by the  
InetAddressDNS textual convention.

Each definition of a concrete InetAddressType value must be  
accompanied by a definition of a textual convention for use  
with that InetAddressType.

To support future extensions, the InetAddressType textual  
convention SHOULD NOT be sub-typed in object type definitions.  
It MAY be sub-typed in compliance statements in order to  
require only a subset of these address types for a compliant  
implementation.

Implementations must ensure that InetAddressType objects

and any dependent objects (e.g. InetAddress objects) are consistent. An inconsistentValue error must be generated if an attempt to change an InetAddressType object would, for example, lead to an undefined InetAddress value. In particular, InetAddressType/InetAddress pairs must be changed together if the address type changes (e.g. from ipv6(2) to ipv4(1))."

```
SYNTAX      INTEGER {
                unknown(0),
                ipv4(1),
                ipv6(2),
                ipv4z(3),
                ipv6z(4),
                dns(16)
            }
```

InetAddress ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"Denotes a generic Internet address.

An InetAddress value is always interpreted within the context of an InetAddressType value. Every usage of the InetAddress textual convention is required to specify the InetAddressType object which provides the context. It is suggested that the InetAddressType object is logically registered before the object(s) which use the InetAddress textual convention if they appear in the same logical row.

The value of an InetAddress object must always be consistent with the value of the associated InetAddressType object. Attempts to set an InetAddress object to a value which is inconsistent with the associated InetAddressType must fail with an inconsistentValue error.

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, STD 58. In this case, the object definition MUST include a 'SIZE' clause to limit the number of potential instance sub-identifiers."

```
SYNTAX      OCTET STRING (SIZE (0..255))
```

InetAddressIPv4 ::= TEXTUAL-CONVENTION

DISPLAY-HINT "1d.1d.1d.1d"

STATUS current

DESCRIPTION

"Represents an IPv4 network address:

octets	contents	encoding
1-4	IPv4 address	network-byte order

The corresponding InetAddressType value is ipv4(1).

This textual convention SHOULD NOT be used directly in object definitions since it restricts addresses to a specific format. However, if it is used, it MAY be used either on its own or in conjunction with InetAddressType as a pair."

SYNTAX OCTET STRING (SIZE (4))

InetAddressIPv6 ::= 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 InetAddressType value is ipv6(2).

This textual convention SHOULD NOT be used directly in object definitions since it restricts addresses to a specific format. However, if it is used, it MAY be used either on its own or in conjunction with InetAddressType as a pair."

SYNTAX OCTET STRING (SIZE (16))

InetAddressIPv4z ::= TEXTUAL-CONVENTION

DISPLAY-HINT "1d.1d.1d.1d%4d"

STATUS current

DESCRIPTION

"Represents a non-global IPv4 network address together with its zone index:

octets	contents	encoding
1-4	IPv4 address	network-byte order
5-8	zone index	network-byte order

The corresponding InetAddressType value is ipv4z(3).

The zone index (bytes 5-8) is used to disambiguate identical address values on nodes which have interfaces attached to different zones of the same scope. The zone index may contain the special value 0 which refers to the default zone for each scope.

This textual convention SHOULD NOT be used directly in object

definitions since it restricts addresses to a specific format. However, if it is used, it MAY be used either on its own or in conjunction with InetAddressType as a pair."

SYNTAX OCTET STRING (SIZE (8))

InetAddressIPv6z ::= TEXTUAL-CONVENTION

DISPLAY-HINT "2x:2x:2x:2x:2x:2x:2x:2x%4d"

STATUS current

DESCRIPTION

"Represents a non-global IPv6 network address together with its zone index:

octets	contents	encoding
1-16	IPv6 address	network-byte order
17-20	zone index	network-byte order

The corresponding InetAddressType value is ipv6z(4).

The zone index (bytes 17-20) is used to disambiguate identical address values on nodes which have interfaces attached to different zones of the same scope. The zone index may contain the special value 0 which refers to the default zone for each scope.

This textual convention SHOULD NOT be used directly in object definitions since it restricts addresses to a specific format. However, if it is used, it MAY be used either on its own or in conjunction with InetAddressType as a pair."

SYNTAX OCTET STRING (SIZE (20))

InetAddressDNS ::= TEXTUAL-CONVENTION

DISPLAY-HINT "255a"

STATUS current

DESCRIPTION

"Represents a DNS domain name. The name SHOULD be fully qualified whenever possible.

The corresponding InetAddressType is dns(16).

The DESCRIPTION clause of InetAddress objects that may have InetAddressDNS values must fully describe how (and when) such names are to be resolved to IP addresses.

This textual convention SHOULD NOT be used directly in object definitions since it restricts addresses to a specific format. However, if it is used, it MAY be used either on its own or in conjunction with InetAddressType as a pair."

SYNTAX OCTET STRING (SIZE (1..255))

InetAddressPrefixLength ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"Denotes the length of a generic Internet network address prefix. A value of n corresponds to an IP address mask which has n contiguous 1-bits from the most significant bit (MSB) and all other bits set to 0.

An InetAddressPrefixLength value is always interpreted within the context of an InetAddressType value. Every usage of the InetAddressPrefixLength textual convention is required to specify the InetAddressType object which provides the context. It is suggested that the InetAddressType object is logically registered before the object(s) which use the InetAddressPrefixLength textual convention if they appear in the same logical row.

InetAddressPrefixLength values that are larger than the maximum length of an IP address for a specific InetAddressType are treated as the maximum significant value applicable for the InetAddressType. The maximum significant value is 32 for the InetAddressType 'ipv4(1)' and 'ipv4z(3)' and 128 for the InetAddressType 'ipv6(2)' and 'ipv6z(4)'. The maximum significant value for the InetAddressType 'dns(16)' is 0.

The value zero is object-specific and must be defined as part of the description of any object which uses this syntax. Examples of the usage of zero might include situations where the Internet network address prefix is unknown or does not apply."

SYNTAX Unsigned32

InetPortNumber ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"Represents a 16 bit port number of an Internet transport layer protocol. Port numbers are assigned by IANA. A current list of all assignments is available from <<http://www.iana.org/>>.

The value zero is object-specific and must be defined as part of the description of any object which uses this syntax. Examples of the usage of zero might include situations where a port number is unknown, or when the value zero is used as a wildcard in a filter."

REFERENCE "STD 6 (RFC 768), STD 7 (RFC 793) and RFC 2960"

SYNTAX Unsigned32 (0..65535)

InetAddressType ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"Represents an autonomous system number which identifies an Autonomous System (AS). An AS is a set of routers under a single technical administration, using an interior gateway protocol and common metrics to route packets within the AS, and using an exterior gateway protocol to route packets to other ASs'. IANA maintains the AS number space and has delegated large parts to the regional registries.

Autonomous system numbers are currently limited to 16 bits (0..65535). There is however work in progress to enlarge the autonomous system number space to 32 bits. This textual convention therefore uses an Unsigned32 value without a range restriction in order to support a larger autonomous system number space."

REFERENCE "RFC 1771, RFC 1930"

SYNTAX Unsigned32

END

#### 4. Usage Hints

The InetAddressType and InetAddress textual conventions have been introduced to avoid over-constraining an object definition by the use of the IPAddress SMI base type which is IPv4 specific. An InetAddressType/InetAddress pair can represent IP addresses in various formats.

The InetAddressType and InetAddress objects SHOULD NOT be sub-typed in object definitions. Sub-typing binds the MIB module to specific address formats, which may cause serious problems if new address formats need to be introduced. Note that it is possible to write compliance statements in order to express that only a subset of the defined address types must be implemented to be compliant.

Every usage of the InetAddress or InetAddressPrefixLength textual conventions must specify which InetAddressType object provides the context for the interpretation of the InetAddress or InetAddressPrefixLength textual convention.

It is suggested that the InetAddressType object is logically registered before the object(s) which uses the InetAddress or InetAddressPrefixLength textual convention. An InetAddressType object is logically registered before an InetAddress or InetAddressPrefixLength object if it appears before the InetAddress or InetAddressPrefixLength object in the conceptual row (which

includes any index objects). This rule allows programs such as MIB compilers to identify the `InetAddressType` of a given `InetAddress` or `InetAddressPrefixLength` object by searching for the `InetAddressType` object which precedes an `InetAddress` or `InetAddressPrefixLength` object.

#### 4.1 Table Indexing

When a generic Internet address is used as an index, both the `InetAddressType` and `InetAddress` objects **MUST** be used. The `InetAddressType` object **MUST** be listed before the `InetAddress` object in the `INDEX` clause.

The `IMPLIED` keyword **MUST NOT** be used for an object of type `InetAddress` in an `INDEX` clause. Instance sub-identifiers are then of the form `T.N.O1.O2...On`, where `T` is the value of the `InetAddressType` object, `O1...On` are the octets in the `InetAddress` 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 lookup specific addresses of known type and value, issue `GetNext` requests for addresses of a single type, or issue `GetNext` requests for a specific type and address prefix.

#### 4.2 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 [19]. 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 an IPv6 address and a zone index is unique [21].

The `InetAddressIPv6` textual convention has been defined to represent global IPv6 addresses and non-global IPv6 addresses in cases where no zone index is needed (e.g., on end hosts with a single interface). The `InetAddressIPv6z` textual convention has been defined to represent non-global IPv6 addresses in cases where a zone index is needed (e.g., a router connecting multiple zones). MIB designers who use `InetAddressType/InetAddress` pairs therefore do not need to define additional objects in order to support non-global addresses on nodes that connect multiple zones.

The `InetAddressIPv4z` is intended for use in MIBs (like the `TCP-MIB`) which report addresses in the address family used on the wire, but where the entity instrumented obtains such addresses from

applications or administrators in a form which includes a zone index, such as v4-mapped IPv6 addresses.

The size of the zone index has been chosen so that it is consistent with (i) the numerical zone index defined in [21] and (ii) the `sin6_scope_id` field of the `sockaddr_in6` structure defined in RFC 2553 [20].

#### 4.3 Multiple Addresses per Host

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 accessible by multiple `InetAddressType/InetAddress` pairs.

If this could be an implementation or usage issue, the `DESCRIPTION` clause of the relevant objects must fully describe which address is reported in a given `InetAddressType/InetAddress` pair.

#### 4.4 Resolving DNS Names

DNS names **MUST** be resolved to IP addresses when communication with the named 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 or just once when the object was instantiated?

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

Similarly, the `DESCRIPTION` clause of such objects **SHOULD** precisely define how and when a reverse lookup is being done if an agent has accessed instrumentation that knows about an IP address and the MIB module or implementation requires it to map the IP address to a DNS name.

#### 5. Table Indexing Example

This example shows a table listing communication peers that are identified by either an IPv4 address, an IPv6 address or a DNS name. The table definition also prohibits entries with an empty address (whose type would be "unknown"). The size of a DNS name is limited to 64 characters in order to satisfy OID length constraints.

```
peerTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF PeerEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of communication peers."
    ::= { somewhere 1 }

peerEntry OBJECT-TYPE
    SYNTAX      PeerEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "An entry containing information about a particular peer."
    INDEX       { peerAddressType, peerAddress }
    ::= { peerTable 1 }

PeerEntry ::= SEQUENCE {
    peerAddressType  InetAddressType,
    peerAddress      InetAddress,
    peerStatus       INTEGER
}

peerAddressType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The type of Internet address by which the peer
         is reachable."
    ::= { peerEntry 1 }

peerAddress OBJECT-TYPE
    SYNTAX      InetAddress (SIZE (1..64))
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The Internet address for the peer. The type of this
         address is determined by the value of the peerAddressType
         object. Note that implementations must limit themselves
         to a single entry in this table per reachable peer.
         The peerAddress may not be empty due to the SIZE
         restriction."
```

If a row is created administratively by an SNMP operation and the address type value is dns(16), then the agent stores the DNS name internally. A DNS name lookup must be performed on the internally stored DNS name whenever it is being used to contact the peer.

If a row is created by the managed entity itself and the address type value is dns(16), then the agent stores the IP address internally. A DNS reverse lookup must be performed on the internally stored IP address whenever the value is retrieved via SNMP."

```
::= { peerEntry 2 }
```

The following compliance statement specifies that compliant implementations need only support IPv4/IPv6 addresses without a zone indices. Support for DNS names or IPv4/IPv6 addresses with zone indices is not required.

```
peerCompliance MODULE-COMPLIANCE
```

```
    STATUS          current
```

```
    DESCRIPTION
```

```
        "The compliance statement of the peer MIB."
```

```
    MODULE          -- this module
```

```
    MANDATORY-GROUPS { peerGroup }
```

```
    OBJECT peerAddressType
```

```
    SYNTAX InetAddressType { ipv4(1), ipv6(2) }
```

```
    DESCRIPTION
```

```
        "An implementation is only required to support IPv4
         and IPv6 addresses without zone indices."
```

```
::= { somewhere 2 }
```

Note that the SMIV2 does not permit inclusion of not-accessible objects in an object group (see section 3.1 in STD 58, RFC 2580 [8]). It is therefore not possible to formally refine the syntax of auxiliary objects which are not-accessible. In such a case, it is suggested to express the refinement informally in the DESCRIPTION clause of the MODULE-COMPLIANCE macro invocation.

## 6. Security Considerations

This module does not define any management objects. Instead, it defines a set of textual conventions which may be used by other MIB modules to define management objects.

Meaningful security considerations can only be written in the MIB modules that define management objects. This document has therefore no impact on the security of the Internet.

## 7. Acknowledgments

This document was produced by the Operations and Management Area "IPv6MIB" design team. The authors would like to thank Fred Baker, Randy Bush, Richard Draves, Mark Ellison, Bill Fenner, Jun-ichiro Hagino, Mike Heard, Tim Jenkins, Glenn Mansfield, Keith McCloghrie, Thomas Narten, Erik Nordmark, Peder Chr. Norgaard, Randy Presuhn, Andrew Smith, Dave Thaler, Kenneth White, Bert Wijnen, and Brian Zill for their comments and suggestions.

## 8. Intellectual Property Notice

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.

## 9. Changes from RFC 2851

The following changes have been made relative to RFC 2851:

- o Added new textual conventions InetAddressPrefixLength, InetPortNumber, and InetAutonomousSystemNumber.

- o Rewrote the introduction to say clearly that in general, one should define MIB tables that work with all versions of IP. The other approach of multiple tables for different IP versions is strongly discouraged.
- o Added text to the InetAddressType and InetAddress descriptions which requires that implementations must reject set operations with an inconsistentValue error if they lead to inconsistencies.
- o Removed the strict ordering constraints. Description clauses now must explain which InetAddressType object provides the context for an InetAddress or InetAddressPrefixLength object.
- o Aligned wordings with the IPv6 scoping architecture document.
- o Split the InetAddressIPv6 textual convention into the two textual conventions (InetAddressIPv6 and InetAddressIPv6z) and introduced a new textual convention InetAddressIPv4z. Added ipv4z(3) and ipv6z(4) named numbers to the InetAddressType enumeration. Motivations for this change: (i) enable the introduction of a textual conventions for non-global IPv4 addresses, (ii) alignment with the textual conventions for transport addresses, (iii) simpler compliance statements in cases where support for IPv6 addresses with zone indices is not required, (iv) simplify implementations for host systems which will never have to report zone indices.

## References

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [2] Harrington, D., Presuhn, R. and B. Wijnen, "An Architecture for Describing SNMP Management Frameworks", RFC 2571, April 1999.
- [3] Rose, M. and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based Internets", STD 16, RFC 1155, May 1990.
- [4] Rose, M. and K. McCloghrie, "Concise MIB Definitions", STD 16, RFC 1212, March 1991.
- [5] Rose, M., "A Convention for Defining Traps for use with the SNMP", RFC 1215, March 1991.
- [6] 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.

- [7] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Textual Conventions for SMIV2", STD 58, RFC 2579, April 1999.
- [8] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Conformance Statements for SMIV2", STD 58, RFC 2580, April 1999.
- [9] Case, J., Fedor, M., Schoffstall, M. and J. Davin, "A Simple Network Management Protocol (SNMP)", STD 15, RFC 1157, May 1990.
- [10] Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Introduction to Community-based SNMPv2", RFC 1901, January 1996.
- [11] 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.
- [12] Case, J., Harrington, D., Presuhn, R. and B. Wijnen, "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)", RFC 2572, April 1999.
- [13] Blumenthal, U. and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", RFC 2574, April 1999.
- [14] 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.
- [15] Levi, D., Meyer, P. and B. Stewart, "SNMP Applications", RFC 2573, April 1999.
- [16] Wijnen, B., Presuhn, R. and K. McCloghrie, "View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)", RFC 2575, April 1999.
- [17] Case, J., Mundy, R., Partain, D. and B. Stewart, "Introduction to Version 3 of the Internet-standard Network Management Framework", RFC 2570, April 1999.
- [18] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, June 2000.
- [19] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 2373, July 1998.

- [20] Gilligan, R., Thomson, S., Bound, J. and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 2553, March 1999.
- [21] Deering, S., Haberman, B., Jinmei, T., Nordmark, E., Onoe, A. and B. Zill, "IPv6 Scoped Address Architecture", Work in Progress.

#### Authors' Addresses

Mike Daniele  
Consultant  
19 Pinewood Rd  
Hudson, NH 03051  
USA

Phone: +1 603 883-6365  
EMail: md@world.std.com

Brian Haberman

Phone: +1 919 949-4828  
EMail: bkhab@nc.rr.com

Shawn A. Routhier  
Wind River Systems, Inc.  
500 Wind River Way  
Alameda, CA 94501  
USA

Phone: +1 510 749 2095  
EMail: sar@epilogue.com

Juergen Schoenwaelder  
TU Braunschweig  
Bueltenweg 74/75  
38106 Braunschweig  
Germany

Phone: +49 531 391-3289  
EMail: schoenw@ibr.cs.tu-bs.de

#### Full Copyright Statement

Copyright (C) The Internet Society (2002). 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.

#### Acknowledgement

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

