

Network Working Group
Internet-Draft
Obsoletes: [3291](#) (if approved)
Expires: February 14, 2005

M. Daniele
Consultant
B. Haberman
Caspian Networks
S. Routhier
Wind River Systems, Inc.
J. Schoenwaelder
International University Bremen
August 16, 2004

Textual Conventions for Internet Network Addresses
draft-ietf-ops-rfc3291bis-06.txt

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which I become aware will be disclosed, in accordance with [RFC 3668](#).

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>.

This Internet-Draft will expire on February 14, 2005.

Copyright Notice

Copyright (C) The Internet Society (2004). 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 will be imported and used in MIB modules that would otherwise define their own representations.

This document obsoletes [RFC 3291](#).

Table of Contents

- [1.](#) Introduction [3](#)
- [2.](#) The Internet-Standard Management Framework [5](#)
- [3.](#) Definitions [5](#)
- [4.](#) Usage Hints [13](#)
 - [4.1](#) Table Indexing [14](#)
 - [4.2](#) Uniqueness of Addresses [14](#)
 - [4.3](#) Multiple Addresses per Host [15](#)
 - [4.4](#) Resolving DNS Names [15](#)
- [5.](#) Table Indexing Example [16](#)
- [6.](#) Security Considerations [18](#)
- [7.](#) IANA Considerations [18](#)
- [8.](#) Acknowledgments [18](#)
- [9.](#) Changes from [RFC 3291](#) to RFC XXXX [18](#)
- [10.](#) Changes from [RFC 2851](#) to [RFC 3291](#) [18](#)
- [11.](#) References [19](#)
 - [11.1](#) Normative References [19](#)
 - [11.2](#) Informative References [20](#)
- Authors' Addresses [20](#)
- Intellectual Property and Copyright Statements [22](#)

1. Introduction

Several standard-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 [[RFC3513](#)].

This document defines multiple textual conventions (TCs) 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 for well-known transport domains support scoped Internet addresses. The scope of an Internet address is a topological span within which the address may be used as a unique identifier for an interface or set of interfaces. A scope zone, or simply a zone, is a concrete connected region of topology of a given scope. Note that a zone is a particular instance of a topological region, whereas a scope is the size of a topological region [[RFCZZZZ](#)]. Since Internet addresses on devices that connect multiple

zones are not necessarily unique, an additional zone index is needed on these devices to select an interface. The textual conventions `InetAddressIPv4z` and `InetAddressIPv6z` are provided to support Internet addresses that include a zone index. In order to support arbitrary combinations of scoped Internet addresses, MIB authors SHOULD use a separate `InetAddressType` object for each `InetAddress` object.

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. A special set of textual conventions for this purpose is defined in [RFC 3419](#) [[RFC3419](#)].

The key words "MUST", "MUST NOT", "SHOULD", "SHOULD NOT" and "MAY" in this document are to be interpreted as described in [RFC 2119](#) [[RFC2119](#)].

2. The Internet-Standard Management Framework

For a detailed overview of the documents that describe the current Internet-Standard Management Framework, please refer to [section 7 of RFC 3410](#) [[RFC3410](#)].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. MIB objects are generally accessed through the Simple Network Management Protocol (SNMP). Objects in the MIB are defined using the mechanisms defined in the Structure of Management Information (SMI). This memo specifies a MIB module that is compliant to the SMIV2, which is described in STD 58,

[RFC 2578](#) [RFC2578], STD 58, [RFC 2579](#) [RFC2579] and STD 58, [RFC 2580](#) [RFC2580].

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 "200408110000Z"
```

```
    ORGANIZATION
```

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

```
    CONTACT-INFO
```

```
        "Juergen Schoenwaelder (Editor)
        International University Bremen
        P.O. Box 750 561
        28725 Bremen, Germany
```

```
        Phone: +49 421 200-3587
```

```
        EMail: j.schoenwaelder@iu-bremen.de
```

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

```
    DESCRIPTION
```

```
        "This MIB module defines textual conventions for
```

Daniele, et al.

Expires February 14, 2005

[Page 5]

Internet-Draft

TCs for Internet Network Addresses

August 2004

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.

Copyright (C) The Internet Society (2004). This version of this MIB module is part of RFC XXXX, see the RFC itself for full legal notices."

```
REVISION "200408110000Z"
```

```
DESCRIPTION
```

```
"Third version, published as RFC XXXX. This revision
introduces the InetZoneIndex, InetScopeType and
```

InetVersion textual conventions."
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) An 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 or else the applicable constraints MUST be stated in the appropriate conceptual row DESCRIPTION clauses or in the surrounding documentation if there is no single DESCRIPTION clause that is appropriate."

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.

The resolution of an InetAddressDNS value may require to query multiple DNS records (e.g., A for IPv4 and AAAA for IPv6). The order of the resolution process and which DNS record takes precedence depends on the configuration of the resolver.

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

DISPLAY-HINT "d"

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.

The upper bound of the prefix length has been chosen to be consistent with the maximum size of an InetAddress."

SYNTAX Unsigned32 (0..2040)

InetPortNumber ::= TEXTUAL-CONVENTION

DISPLAY-HINT "d"

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)

InetAutonomousSystemNumber ::= TEXTUAL-CONVENTION

DISPLAY-HINT "d"

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.

Daniele, et al.

Expires February 14, 2005

[Page 11]

Internet-Draft

TCs for Internet Network Addresses

August 2004

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

InetScopeType ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"Represents a scope type. This textual convention can be used in cases where a MIB has to represent different scope types and there is no context information such as an InetAddress object which implicitly defines the scope type.

Note that not all possible values have been assigned yet but they may be assigned in future revisions of this specification. Applications should therefore be able to deal with not yet assigned values."

REFERENCE "[RFC 3513](#)"

```

SYNTAX      INTEGER {
                -- reserved(0),
                interfaceLocal(1),
                linkLocal(2),
                subnetLocal(3),
                adminLocal(4),
                siteLocal(5),
                -- unassigned(6),
                -- unassigned(7),
                organizationLocal(8),
                -- unassigned(9),
                -- unassigned(10),
                -- unassigned(11),
                -- unassigned(12),
                -- unassigned(13),
                global(14)
                -- reserved(15)
            }

```

InetZoneIndex ::= TEXTUAL-CONVENTION

DISPLAY-HINT "d"

STATUS current

DESCRIPTION

"A zone index identifies an instance of a zone of a specific scope."

The zone index MUST disambiguate identical address values. For link-local addresses, the zone index will typically be the interface index (ifIndex as defined in the IF-MIB) of the interface on which the address is configured.

The zone index may contain the special value 0 which refers to the default zone. The default zone may be used in cases where the valid zone index is not known (e.g., a management application needs to write a link-local IPv6 address without knowing the interface index value). The default zone SHOULD NOT be used as an easy way out in cases where the zone index for a non-global IPv6 address is known."

REFERENCE "RFCZZZZ"

SYNTAX Unsigned32

```

InetVersion ::= TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
        "A value representing a version of the IP protocol.

        unknown(0)  An unknown or unspecified version of the IP
                    protocol.

        ipv4(1)     The IPv4 protocol as defined in RFC 791 (STD 5).

        ipv6(2)     The IPv6 protocol as defined in RFC 2460.

        Note that this textual convention SHOULD NOT be used to
        distinguish different address types associated with IP
        protocols. The InetAddressType has been designed for this
        purpose."
    REFERENCE      "RFC 791, RFC 2460"
    SYNTAX         INTEGER {
                    unknown(0),
                    ipv4(1),
                    ipv6(2)
                }
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.01.02...0n, where T is the value of the InetAddressType object, 01...0n 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 [[RFC3513](#)]. In particular, IPv6

"link-local" unicast 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 [[RFCZZZZ](#)].

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 [[RFCZZZZ](#)] and (ii) the `sin6_scope_id` field of the `sockaddr_in6` structure defined in [RFC 2553](#) [[RFC2553](#)].

[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 }

Daniele, et al.

Expires February 14, 2005

[Page 17]

Internet-Draft

TCs for Internet Network Addresses

August 2004

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](#) [[RFC2580](#)]). 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.](#) IANA Considerations

This document has no actions for IANA.

[8.](#) 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, Allison Mankin, 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.

[9.](#) Changes from [RFC 3291](#) to RFC XXXX

The following changes have been made relative to [RFC 3291](#):

- o Added a range restriction to the InetAddressPrefixLength textual convention.
- o Added new textual conventions InetZoneIndex, InetScopeType and InetVersion.
- o Added explicit "d" DISPLAY-HINTs for textual conventions that did not have them.
- o Updated boilerplate text and references.

[10.](#) Changes from [RFC 2851](#) to [RFC 3291](#)

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.

[11.](#) References

11.1 Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC2578] McCloghrie, K., Perkins, D. and J. Schoenwaelder, "Structure of Management Information Version 2 (SMIv2)", STD 58, [RFC 2578](#), April 1999.
- [RFC2579] McCloghrie, K., Perkins, D. and J. Schoenwaelder, "Textual Conventions for SMIv2", STD 58, [RFC 2579](#), April 1999.
- [RFC2580] McCloghrie, K., Perkins, D. and J. Schoenwaelder, "Conformance Statements for SMIv2", STD 58, [RFC 2580](#), April 1999.
- [RFC3513] Hinden, R. and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture", [RFC 3513](#), April 2003.
- [RFCZZZZ] Deering, S., Haberman, B., Jinmei, T., Nordmark, E. and B. Zill, "IPv6 Scoped Address Architecture", [draft-ietf-ipv6-scoping-arch-01.txt](#), February 2004.

Daniele, et al.

Expires February 14, 2005

[Page 19]

Internet-Draft

TCs for Internet Network Addresses

August 2004

11.2 Informative References

- [RFC3410] Case, J., Mundy, R., Partain, D. and B. Stewart, "Introduction and Applicability Statements for the Internet-Standard Management Framework", [RFC 3410](#), December 2002.
- [RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", [RFC 2863](#), June 2000.
- [RFC2553] Gilligan, R., Thomson, S., Bound, J. and W. Stevens, "Basic Socket Interface Extensions for IPv6", [RFC 2553](#), March 1999.
- [RFC3419] Daniele, M. and J. Schoenwaelder, "Textual Conventions for Transport Addresses", [RFC 3419](#), December 2002.

Authors' Addresses

Mike Daniele
Consultant
19 Pinewood Rd
Hudson, NH 03051
USA

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

Brian Haberman
Caspian Networks
1 Park Drive, Suite 300
Research Triangle Park, NC 27709
USA

Phone: +1 919-949-4828
EMail: brian@innovationslab.net

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

Phone: +1 510 749-2095
EMail: shawn.routhier@windriver.com

Daniele, et al.

Expires February 14, 2005

[Page 20]

Internet-Draft

TCs for Internet Network Addresses

August 2004

Juergen Schoenwaelder
International University Bremen
P.O. Box 750 561
28725 Bremen
Germany

Phone: +49 421 200-3587
EMail: j.schoenwaelder@iu-bremen.de

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights 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; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in [BCP 78](#) and [BCP 79](#).

Copies of IPR disclosures made to the IETF Secretariat 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 implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM 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.

Copyright Statement

Copyright (C) The Internet Society (2004). This document is subject to the rights, licenses and restrictions contained in [BCP 78](#), and except as set forth therein, the authors retain all their rights.

Acknowledgment

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