

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

M. Daniele
Compaq Computer Corporation
B. Haberman
Nortel Networks
S. Routhier
Wind River Systems, Inc.
J. Schoenwaelder
TU Braunschweig
June 2000

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 (2000). All Rights Reserved.

Abstract

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

This work is output from the Operations and Management Area "IPv6MIB" design team.

Table of Contents

1. Introduction	2
2. The SNMP Management Framework	3
3. Definitions	4
4. Usage Hints	8
4.1 Table Indexing	8
4.2 Uniqueness of Addresses	9
4.3 Multiple InetAddresses per Host	9
4.4 Resolving DNS Names	9
5. Table Indexing Example	10
6. Security Considerations	12
7. Acknowledgments	12

8. Intellectual Property Notice	12
References	13
Authors' Addresses	15
Full Copyright Statement	16

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 [21].

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 MIBs 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 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 MIBs that only need to represent IPv4 or IPv6 addresses SHOULD use the textual conventions defined in this memo.

In order to make existing widely-deployed IPv4-only MIBs fit for IPv6, it might be a valid approach to define separate tables for different address types. This is a decision for the MIB designer. For example, the tcpConnTable of the TCP-MIB [18] was left intact

and a new table was added for TCP connections over IPv6 in the IPV6-TCP-MIB [19]. Note that even in this case, the MIBs SHOULD use the textual conventions defined in this memo.

Note that MIB developers SHOULD NOT use the textual conventions defined in this document to represent 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 FROM SNMPv2-SMI  
    TEXTUAL-CONVENTION      FROM SNMPv2-TC;
```

```
inetAddressMIB MODULE-IDENTITY
```

```
    LAST-UPDATED "200006080000Z"
```

```
    ORGANIZATION
```

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

```
    CONTACT-INFO
```

```
        "Mike Daniele  
        Compaq Computer Corporation  
        110 Spit Brook Rd  
        Nashua, NH 03062, USA
```

```
        Phone: +1 603 884-1423  
        EMail: danielle@zk3.dec.com
```

```
        Brian Haberman  
        Nortel Networks  
        4039 Emperor Blvd., Suite 200  
        Durham, NC 27703, USA
```

```
        Phone: +1 919 992-4439  
        EMail: haberman@nortelnetworks.com
```

```
        Shawn A. Routhier  
        Wind River Systems, Inc.  
        1 Tara Blvd, Suite 403  
        Nashua, NH 03062, USA
```

```
        Phone: +1 603 897-2000  
        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

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."

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.

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.

The InetAddressType textual convention SHOULD NOT be subtyped in object type definitions to support future extensions. It

MAY be subtyped in compliance statements in order to require only a subset of these address types for a compliant implementation."

```
SYNTAX      INTEGER {
                unknown(0),
                ipv4(1),    -- these named numbers are aligned
                ipv6(2),    -- with AddressFamilyNumbers from
                dns(16)     -- IANA-ADDRESS-FAMILY-NUMBERS-MIB
            }
```

InetAddress ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"Denotes a generic Internet address.

An InetAddress value is always interpreted within the context of an InetAddressType value. The InetAddressType object which defines the context must be registered immediately before the object which uses the InetAddress textual convention. In other words, the object identifiers for the InetAddressType object and the InetAddress object MUST have the same length and the last sub-identifier of the InetAddressType object MUST be 1 less than the last sub-identifier of the InetAddress object.

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-TYPE declaration 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	IP address	network-byte order

The corresponding InetAddressType value is ipv4(1)."

```
SYNTAX      OCTET STRING (SIZE (4))
```

InetAddressIPv6 ::= TEXTUAL-CONVENTION

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

STATUS current

DESCRIPTION

"Represents an IPv6 network address:

octets	contents	encoding
1-16	IPv6 address	network-byte order
17-20	scope identifier	network-byte order

The corresponding InetAddressType value is ipv6(2).

The scope identifier (bytes 17-20) MUST NOT be present for global IPv6 addresses. For non-global IPv6 addresses (e.g. link-local or site-local addresses), the scope identifier MUST always be present. It contains a link identifier for link-local and a site identifier for site-local IPv6 addresses.

The scope identifier MUST disambiguate identical address values. For link-local addresses, the scope identifier will typically be the interface index (ifIndex as defined in the IF-MIB, RFC 2233) of the interface on which the address is configured.

The scope identifier may contain the special value 0 which refers to the default scope. The default scope may be used in cases where the valid scope identifier is not known (e.g., a management application needs to write a site-local InetAddressIPv6 address without knowing the site identifier value). The default scope SHOULD NOT be used as an easy way out in cases where the scope identifier for a non-global IPv6 is known."

SYNTAX OCTET STRING (SIZE (16|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."

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

END

4. Usage Hints

One particular usage of `InetAddressType/InetAddress` pairs is to avoid over-constraining an object definition by the use of the `IpAddress` SMI base type. An `InetAddressType/InetAddress` pair allows to represent IP addresses in various formats.

The `InetAddressType` and `InetAddress` objects SHOULD NOT be subtyped. Subtyping 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.

Internet addresses MUST always be represented by a pair of `InetAddressType/InetAddress` objects. It is not allowed to "share" an `InetAddressType` between multiple `InetAddress` objects. Furthermore, the `InetAddressType` object must be registered immediately before the `InetAddress` object. In other words, the object identifiers for the `InetAddressType` object and the `InetAddress` object MUST have the same length and the last sub-identifier of the `InetAddressType` object MUST be 1 less than the last sub-identifier of the `InetAddress` 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 come immediately before the `InetAddress` object in the INDEX clause. If multiple Internet addresses are used in the INDEX clause, then every Internet address must be represented by a pair of `InetAddressType` and `InetAddress` objects.

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...On`, where `T` is the value of the `InetAddressType` object, `01...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 [21]. 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 an interface number is unique. The interface number may therefore be used as a scope identifier.

The InetAddressIPv6 textual convention has been defined to represent global and non-global IPv6 addresses. MIB designers who use InetAddressType/InetAddress pairs therefore do not need define additional objects in order to support link-local or site-local addresses.

The size of the scope identifier has been chosen so that it matches the `sin6_scope_id` field of the `sockaddr_in6` structure defined in RFC 2553 [22].

4.3 Multiple InetAddresses 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 represented by multiple InetAddressType/InetAddress pairs.

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

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? 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 or implementation requires to map the address to a 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.

```
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. 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 implementations need only support IPv4 addresses and globally unique IPv6 addresses to be compliant. Support for DNS names or scoped IPv6 addresses is not required.

peerCompliance MODULE-COMPLIANCE

STATUS current

DESCRIPTION

"The compliance statement 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."

OBJECT peerAddress

SYNTAX InetAddress (SIZE(4|16))

DESCRIPTION

"An implementation is only required to support IPv4 and globally unique IPv6 addresses."

```
::= { 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 modules that define management objects.

7. Acknowledgments

The authors would like to thank Randy Bush, Richard Draves, Mark Ellison, Bill Fenner, Jun-ichiro Hagino, 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.

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., "SNMPv2 Management Information Base for the Transmission Control Protocol using SMIV2", RFC 2012, November 1996.
- [19] Daniele, M., "IP Version 6 Management Information Base for the Transmission Control Protocol", RFC 2452, December 1998.
- [20] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB using SMIV2", RFC 2233, November 1997.
- [21] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 2373, July 1998.
- [22] Gilligan, R., Thomson, S., Bound, J. and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 2553, March 1999.

Authors' Addresses

Mike Daniele
Compaq Computer Corporation
110 Spit Brook Rd
Nashua, NH 03062
USA

Phone: +1 603 884-1423
EMail: daniele@zk3.dec.com

Brian Haberman
Nortel Networks
4039 Emperor Blvd., Suite 200
Durham, NC 27703
USA

Phone: +1 919 992-4439
EMail: haberman@nortelnetworks.com

Shawn A. Routhier
Wind River Systems, Inc.
1 Tara Blvd, Suite 403
Nashua, NH 03062
USA

Phone: +1 603 897-2000
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 (2000). 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.

