IPv6 MIB Revision Design Team   Bill Fenner
INTERNET-DRAFT   AT&T Research
Expires: January 2003   Editor
                                                                 July 2002

Management Information Base
for the User Datagram Protocol (UDP)
draft-ietf-ipv6-rfc2013-update-00.txt

Status of this Document

This document is a product of the IPv6 MIB Revision Design Team. Comments should be addressed to the authors, or the mailing list at ipng@sunroof.eng.sun.com.

Copyright Notice

Abstract

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols in the Internet community. In particular, it describes managed objects used for implementations of the User Datagram Protocol (UDP) [4] in an IP version independent manner.

It is intended to obsolete RFC 2013 and RFC 2454.

Table of Contents

1. The SNMP Management Framework

The SNMP Management Framework presently consists of five major components:

- An overall architecture, described in RFC 2571 [5].

- 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 [6], STD 16, RFC 1212 [7] and RFC 1215 [8]. The second version, called SMIv2, is described in STD 58, RFC 2578 [9], STD 58, RFC 2579 [10] and STD 58, RFC 2580 [11].

- Message protocols for transferring management information. The first version of the SNMP message protocol is called SNMPv1 and described in STD 15, RFC 1157 [12]. A second version of the SNMP message protocol, which is not an Internet standards track protocol, is called SNMPv2c and described in RFC 1901 [13] and RFC 1906 [14]. The third version of the message protocol is called SNMPv3 and described in RFC 1906 [14], RFC 2572 [15] and RFC 2574 [16].

- Protocol operations for accessing management information. The first set of protocol operations and associated PDU formats is described in STD 15, RFC 1157 [12]. A second set of protocol operations and associated PDU formats is described in RFC 1905 [17].

o A set of fundamental applications described in RFC 2573 [18] and the view-based access control mechanism described in RFC 2575 [19].

A more detailed introduction to the current SNMP Management Framework can be found in RFC 2570 [20].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are defined using the mechanisms defined in the SMI.

This memo specifies a MIB module that is compliant to the SMIv2. A MIB conforming to the SMIv1 can be produced through the appropriate translations. The resulting translated MIB must be semantically equivalent, except where objects or events are omitted because no translation is possible (use of Counter64). Some machine readable information in SMIv2 will be converted into textual descriptions in SMIv1 during the translation process. However, this loss of machine readable information is not considered to change the semantics of the MIB.

2. Revision History

Changes from draft-ietf-ipngwg-rfc2013-update-01.txt

28 May 2002

Removed udpConnectionTable

Renamed ListenerTable to EndpointTable, since with a remote address Listener is not quite correct.

Use ''h consistently for 'any IP address', instead of sometimes ''h and sometimes all-zeroes of the right address family.

Use "Datagram" instead of "Packet" to talk about UDP datagrams.

Added mandatory udpEndpointStartTime, this also fixes the udpEndpointInstance needing to be read-only and mandatory.

Make udpEndpointProcess mandatory on systems that have process IDs.

Make a note of { udp 6 } in a comment for clarity on why it's skipped.

Fleshed out section 3.

Changed the deprecated udpLocalPort SYNTAX to Integer32. Since it was already restricted to (0..65536) this is not a semantic change.

Changes from draft-ietf-ipngwg-rfc2013-update-00.txt

14 November 2001

Added udpConnectionTable

Added udpListenerRemoteAddressType, to distinguish e.g. IPV6_V6ONLY

Added counters to udpListenerTable and udpConnectionTable

Changes from draft-ops-rfc2013-update-00.txt

12 Jul 2001

Turned into IPNG WG document

Changes from first draft posted to v6mib mailing list:

23 Feb 2001

Made threshold for HC packet counters 1Mpps

Added copyright statements and table of contents

21 Feb 2001 -- Juergen's changes

Renamed udpInetTable to udpListenerTable

Updated Conformance info

6 Feb 2001

Removed v6-only objects.

Removed remote and instance objects, turning the table back into a listener-only table.

Renamed inetUdp* to udpInet*

Added HC in and out datagram counters

Added SIZE restriction to udpListenerLocalAddress. (36 = 32-byte addresses plus 4-byte scope, but it's just a strawman)

Used InetPortNumber TC from updated INET-ADDRESS-MIB

Updated compliance statements.

Added Keith to authors

Added open issues section.

3. MIB Structure

The current UDP-MIB defined in this memo consists of one tables an a group of scalars:

- The udp group of scalars reports parameters and statistics of a UDP protocol engine. Two scalars udpHCInDatagrams and udpHCOutDatagrams have been added to this group since the publication of RFC 2013 in order to provide high-capacity counters for fast networks.

- The udpEndpointTable provides access to status information for all UDP endpoints handled by a UDP protocol engine. The table provides for strictly listening endpoints, as with the historical udpTable, and also for "connected" UDP endpoints, which only accept packets from a given remote system. It also reports identification of the operating system level processes which handles UDP connections and the start time of a connection.

3.1. Relationship to Other MIBs

This section discusses the relationship of this UDP-MIB module to other MIB modules.

3.1.1. Relationship to RFC1213-MIB

UDP related MIB objects were originally defined as part of the RFC1213-MIB defined in RFC 1213. The UDP related objects of the RFC1213-MIB were later copied into a separate MIB module and published in RFC 2013 in SMIv2 format.

The previous versions of the UDP-MIB both defined the udpTable, which has been deprecated for basically two reasons:

(1) The udpTable only supports IPv4.

The current approach in the IETF is to write IP version neutral MIBs rather than having different definitions for various version of IP. This reduces the amount of overhead when new objects are introduced since there is only one place to add them. Hence, the approach taken in RFC 2453 of having separate tables is not continued.

(2) The udpTable does not permit describing "connected" UDP endpoints.

It turns out that "connected" endpoints tend to have a different behaviour and management access pattern compared to listening endpoints. Adding remote endpoint information to the udpEndpointTable thus allows to add specific status and statistic objects for "connected" endpoints and connections.

3.1.2. Relationship to the IPV6-UDP-MIB

The IPV6-UDP-MIB defined in RFC 2453 has been moved to Historic since the approach of having separate IP version specific tables is not followed anymore. Implementation of RFC 2453 is thus not suggested anymore.

3.1.3. Relationship to HOST-RESOURCES-MIB and SYSAPPL-MIB

The udpEndpointTable reports the identification of the operating system level process which handles a connection or a listening endpoint. The value is reported as an Unsigned32 which is expected to be the same as the hrSWRunIndex of the HOST-RESOURCES-MIB (RFC 2790) (if the value is smaller than 2147483647) or the sysApplElmtRunIndex of the SYSAPPL-MIB (RFC 2287). This allows managment applications to identify the UDP connections that belong to an operating system level process which has proven to be valuable in operational environments.

3.2. Use of IP Addresses

(this section needs to be fleshed out)

Addresses are as seen on the wire, not necessarily as the socket sees them (e.g. IPv4 address, not IPv6-mapped IPv4).

Listener on in6addr_any without IPV6_V6ONLY socket option set (i.e. willing to accept v4 or v6) is indicated by AF = unknown.

4. Definitions

```
UDP-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, Counter32, Counter64, Unsigned32,
    Integer32, IpAddress, mib-2        FROM SNMPv2-SMI
    TimeStamp                          FROM SNMPv2-TC
    MODULE-COMPLIANCE, OBJECT-GROUP    FROM SNMPv2-CONF
    InetAddress, InetAddressType,
    InetPortNumber                     FROM INET-ADDRESS-MIB;
```

```
udpMIB MODULE-IDENTITY
    LAST-UPDATED "200111150000Z"
    ORGANIZATION "IETF IPv6 MIB Revision Team"
    CONTACT-INFO
           "Bill Fenner (editor)

            AT&T Labs -- Research
            75 Willow Rd.
            Menlo Park, CA 94025

            Phone: +1 650 330-7893
            Email: <fenner@research.att.com>"
    DESCRIPTION
           "The MIB module for managing UDP implementations."
    REVISION      "200111150000Z"
    DESCRIPTION
           "IP version neutral revision, published as RFC XXXX."
    REVISION      "9411010000Z"
    DESCRIPTION
           "Initial SMIv2 version, published as RFC 2013."
    REVISION      "9103310000Z"
    DESCRIPTION
           "The initial revision of this MIB module was part of MIB-II."
    ::= { mib-2 50 }

-- the UDP group

udp      OBJECT IDENTIFIER ::= { mib-2 7 }

udpInDatagrams OBJECT-TYPE
    SYNTAX     Counter32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The total number of UDP datagrams delivered to UDP users."
    ::= { udp 1 }

udpNoPorts OBJECT-TYPE
    SYNTAX     Counter32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The total number of received UDP datagrams for which there
            was no application at the destination port."
    ::= { udp 2 }

udpInErrors OBJECT-TYPE
    SYNTAX     Counter32
```

```
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The number of received UDP datagrams that could not be
            delivered for reasons other than the lack of an application
            at the destination port."
    ::= { udp 3 }

udpOutDatagrams OBJECT-TYPE
    SYNTAX     Counter32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The total number of UDP datagrams sent from this entity."
    ::= { udp 4 }

udpHCInDatagrams OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The total number of UDP datagrams delivered to UDP users,
            for devices which can receive more than 1 million UDP
            datagrams per second."
    ::= { udp 8 }

udpHCOutDatagrams OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The total number of UDP datagrams sent from this entity, for
            devices which can transmit more than 1 million UDP datagrams
            per second."
    ::= { udp 9 }

--
-- { udp 6 } was defined as the ipv6UdpTable in RFC2454's IPV6-UDP-MIB.
-- This RFC obsoletes RFC 2454, so { udp 6 } is obsoleted.
--

-- The UDP "Endpoint" table.

udpEndpointTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF UdpEndpointEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
```

```
           "A table containing information about this entity's UDP
            endpoints on which a local application is currently
            accepting or sending datagrams.

            The address type in this table represents the address type
            used for the communication, irrespective of the higher-layer
            abstraction.  For example, an application using IPv6
            'sockets' to communicate via IPv4 between ::ffff:10.0.0.1
            and ::ffff:10.0.0.2 would use InetAddressType ipv4(1).

            Unlike the udpTable in RFC 2013, this table also allows the
            representation of an application which completely specifies
            both local and remote addresses and ports.  A listening
            application is represented in three possible ways:

            1) an application which is willing to accept both IPv4 and
            IPv6 datagrams is represented by a
            udpEndpointLocalAddressType of unknown(0) and
            udpEndpointLocalAddress and udpEndpointRemoteAddress of ''h
            (a zero-length octet-string).

            2) an application which is willing to accept only IPv4 or
            only IPv6 datagrams is represented by a
            udpEndpointLocalAddressType of the appropriate address type,
            and udpEndpointLocalAddress and udpEndpointRemoteAddress of
            ''h (a zero-length octet-string).

            3) an application which is listening for datagrams only for
            a specific IP address, but from any remote system, is
            repesented by a udpEndpointLocalAddressType of the
            appropriate address type, udpEndpointLocalAddress specifying
            the local address, and udpEndpointRemoteAddress of ''h (a
            zero-length octet-string).

            In all cases where the remote is a wildcard, the
            udpEndpointRemotePort is 0.

            If the operating system is demultiplexing UDP packets by
            remote address and port, or if the application has
            'connected' the socket specifying a default remote address
            and port, the udpEndpointRemote* values should be used to
            reflect this."
    ::= { udp 7 }

udpEndpointEntry OBJECT-TYPE
    SYNTAX     UdpEndpointEntry
    MAX-ACCESS not-accessible
    STATUS     current
```

```
    DESCRIPTION
           "Information about a particular current UDP endpoint."
    INDEX   { udpEndpointLocalAddressType,
              udpEndpointLocalAddress,
              udpEndpointLocalPort,
              udpEndpointRemoteAddress,
              udpEndpointRemotePort,
              udpEndpointInstance }
    ::= { udpEndpointTable 1 }

UdpEndpointEntry ::= SEQUENCE {
        udpEndpointLocalAddressType  InetAddressType,
        udpEndpointLocalAddress      InetAddress,
        udpEndpointLocalPort         InetPortNumber,
        udpEndpointRemoteAddress     InetAddress,
        udpEndpointRemotePort        InetPortNumber,
        udpEndpointInstance          Unsigned32,
        udpEndpointInDatagrams       Counter32,
        udpEndpointHCInDatagrams     Counter64,
        udpEndpointOutDatagrams      Counter32,
        udpEndpointHCOutDatagrams    Counter64,
        udpEndpointInOctets          Counter32,
        udpEndpointHCInOctets        Counter64,
        udpEndpointOutOctets         Counter32,
        udpEndpointHCOutOctets       Counter64,
        udpEndpointStartTime         TimeStamp,
        udpEndpointProcess           Unsigned32
    }


udpEndpointLocalAddressType OBJECT-TYPE
    SYNTAX     InetAddressType
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "The address type of udpEndpointLocalAddress.  Only IPv4 and
            IPv6 addresses are expected, or unknown(0) if datagrams for
            all local IP addresses are accepted."
    ::= { udpEndpointEntry 1 }

udpEndpointLocalAddress OBJECT-TYPE
    SYNTAX     InetAddress (SIZE(0..36))
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "The local IP address for this UDP endpoint.  This is either
            one of the IP addresses assigned to the system, or a null
            octet-string (''h) to represent that datagrams destined to
```

```
            any address assigned to the system of an IP version
            consistent with udpEndpointLocalAddressType (or any IP
            version, if udpEndpointLocalAddressType is unknown(0)) will
            be accepted."
    ::= { udpEndpointEntry 2 }

udpEndpointLocalPort OBJECT-TYPE
    SYNTAX     InetPortNumber
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "The local port number for this UDP endpoint."
    ::= { udpEndpointEntry 3 }

udpEndpointRemoteAddress OBJECT-TYPE
    SYNTAX     InetAddress (SIZE(0..36))
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "The remote IP address for this UDP endpoint.  If datagrams
            from any remote system are to be accepted, this value is ''h
            (a zero-length octet-string).  Otherwise, it has the type
            described by udpEndpointLocalAddress, and is the address of
            the remote system from which datagrams are to be accepted
            (or to which all datagrams will be sent)."
    ::= { udpEndpointEntry 4 }

udpEndpointRemotePort OBJECT-TYPE
    SYNTAX     InetPortNumber
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "The remote port number for this UDP endpoint.  If datagrams
            from any remote system are to be accepted, this value is
            zero."
    ::= { udpEndpointEntry 5 }

udpEndpointInstance OBJECT-TYPE
    SYNTAX     Unsigned32 (1..'ffffffff'h)
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "The instance of this tuple.  This object is used to
            distinguish between multiple processes 'connected' to the
            same UDP endpoint."
    ::= { udpEndpointEntry 6 }

udpEndpointInDatagrams OBJECT-TYPE
```

```
    SYNTAX     Counter32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The count of datagrams received for this endpoint."
    ::= { udpEndpointEntry 7 }

udpEndpointHCInDatagrams OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The count of datagrams received for this endpoint, for
            devices which can receive more than 1 million UDP datagrams
            per second."
    ::= { udpEndpointEntry 8 }

udpEndpointOutDatagrams OBJECT-TYPE
    SYNTAX     Counter32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The count of datagrams sent on this endpoint."
    ::= { udpEndpointEntry 9 }

udpEndpointHCOutDatagrams OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The count of datagrams sent on this endpoint, for devices
            which can transmit more than 1 million UDP datagrams per
            second."
    ::= { udpEndpointEntry 10 }

udpEndpointInOctets OBJECT-TYPE
    SYNTAX     Counter32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The count of octets received for this endpoint."
    ::= { udpEndpointEntry 11 }

udpEndpointHCInOctets OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
```

```
           "The count of octets received for this endpoint, for devices
            which can receive more than 1 million UDP octets per
            second."
    ::= { udpEndpointEntry 12 }

udpEndpointOutOctets OBJECT-TYPE
    SYNTAX     Counter32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The count of octets sent on this endpoint."
    ::= { udpEndpointEntry 13 }

udpEndpointHCOutOctets OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The count of octets sent on this endpoint, for devices which
            can transmit more than 1 million UDP octets per second."
    ::= { udpEndpointEntry 14 }

udpEndpointStartTime OBJECT-TYPE
    SYNTAX     TimeStamp
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The value of sysUpTime at the time this endpoint was
            established."
    ::= { udpEndpointEntry 15 }

udpEndpointProcess OBJECT-TYPE
    SYNTAX     Unsigned32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The system's process ID for the process associated with this
            endpoint, or zero if there is no such process.  This value
            is expected to be the same as HOST-RESOURCES-
            MIB::hrSWRunIndex or SYSAPPL-MIB::sysApplElmtRunIndex for
            some row in the appropriate tables."
    ::= { udpEndpointEntry 16 }



-- The deprecated UDP Listener table

-- The deprecated UDP listener table only contains information about this
```

```
-- entity's IPv4 UDP end-points on which a local application is
-- currently accepting datagrams.  It does not provide more detailed
-- connection information, or information about IPv6 endpoints.

udpTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF UdpEntry
    MAX-ACCESS not-accessible
    STATUS     deprecated
    DESCRIPTION
           "A table containing IPv4-specific UDP listener information.
            It contains information about all local IPv4 UDP end-points
            on which an application is currently accepting datagrams.
            This table has been deprecated in favor of the version
            neutral udpEndpointTable."
    ::= { udp 5 }

udpEntry OBJECT-TYPE
    SYNTAX     UdpEntry
    MAX-ACCESS not-accessible
    STATUS     deprecated
    DESCRIPTION
           "Information about a particular current UDP listener."
    INDEX   { udpLocalAddress, udpLocalPort }
    ::= { udpTable 1 }

UdpEntry ::= SEQUENCE {
        udpLocalAddress  IpAddress,
        udpLocalPort     Integer32
    }


udpLocalAddress OBJECT-TYPE
    SYNTAX     IpAddress
    MAX-ACCESS read-only
    STATUS     deprecated
    DESCRIPTION
           "The local IP address for this UDP listener.  In the case of
            a UDP listener which is willing to accept datagrams for any
            IP interface associated with the node, the value 0.0.0.0 is
            used."
    ::= { udpEntry 1 }

udpLocalPort OBJECT-TYPE
    SYNTAX     Integer32 (0..65535)
    MAX-ACCESS read-only
    STATUS     deprecated
    DESCRIPTION
           "The local port number for this UDP listener."
```

```
    ::= { udpEntry 2 }



-- conformance information

udpMIBConformance OBJECT IDENTIFIER ::= { udpMIB 2 }

udpMIBCompliances OBJECT IDENTIFIER ::= { udpMIBConformance 1 }
udpMIBGroups      OBJECT IDENTIFIER ::= { udpMIBConformance 2 }

-- compliance statements

udpMIBCompliance2 MODULE-COMPLIANCE
    STATUS     current
    DESCRIPTION
           "The compliance statement for systems which implement UDP."
    MODULE  -- this module
         MANDATORY-GROUPS { udpBaseGroup, udpEndpointGroup }
         GROUP       udpHCGroup
         DESCRIPTION
           "This group is mandatory for those systems which are capable
                 of receiving or transmitting more than 1 million UDP
                 datagrams per second.  1 million datagrams per second
                 will cause a Counter32 to wrap in just over an hour."
         GROUP       udpEndpointProcessGroup
         DESCRIPTION
           "This group is mandatory for systems which implement a
                 'process ID' concept, in particular those that also
                 implement the HOST-RESOURCES-MIB or SYSAPPL-MIB."
         GROUP       udpEndpointStatsGroup
         DESCRIPTION
           "This group is optional."
         GROUP       udpEndpointHCDatagramStatsGroup
         DESCRIPTION
           "This group is mandatory for systems which implement
                 udpEndpointStatsGroup and are capable of receiving or
                 transmitting more than 1 million UDP datagrams per
                 second.  1 million datagrams per second will cause a
                 Counter32 to wrap in just over an hour."
         GROUP       udpEndpointHCOctetStatsGroup
         DESCRIPTION
           "This group is mandatory for systems which implement
                 udpEndpointStatsGroup and are capable of receiving or
                 transmitting more than 1 million UDP octets per second
                 (approximately 15 full-sized IP packets per second).  1
                 million octets per second will cause a Counter32 to
                 wrap in just over an hour."
```

```
    ::= { udpMIBCompliances 2 }

udpMIBCompliance MODULE-COMPLIANCE     STATUS     deprecated
    DESCRIPTION
           "The compliance statement for IPv4-only systems which
            implement UDP.  For IP version independence, this compliance
            statement is deprecated in favor of udpMIBCompliance2.
            However, agents are still encouraged to implement these
            objects in order to interoperate with the deployed base of
            managers."
    MODULE  -- this module
        MANDATORY-GROUPS { udpGroup }
    ::= { udpMIBCompliances 1 }

-- units of conformance

udpGroup OBJECT-GROUP
    OBJECTS   { udpInDatagrams, udpNoPorts,
                udpInErrors, udpOutDatagrams,
                udpLocalAddress, udpLocalPort }
    STATUS     deprecated
    DESCRIPTION
           "The deprecated group of objects providing for management of
            UDP over IPv4."
    ::= { udpMIBGroups 1 }

udpBaseGroup OBJECT-GROUP
    OBJECTS   { udpInDatagrams, udpNoPorts, udpInErrors, udpOutDatagrams }
    STATUS     current
    DESCRIPTION
           "The group of objects providing for counters of UDP
            statistics."
    ::= { udpMIBGroups 2 }

udpHCGroup OBJECT-GROUP
    OBJECTS   { udpHCInDatagrams, udpHCOutDatagrams }
    STATUS     current
    DESCRIPTION
           "The group of objects providing for counters of very high
            speed UDP implementations."
    ::= { udpMIBGroups 3 }

udpEndpointGroup OBJECT-GROUP
    OBJECTS    { udpEndpointStartTime }
    STATUS     current
    DESCRIPTION
           "The group of objects providing for the IP version
            independent management of UDP 'endpoints'."
```

```
    ::= { udpMIBGroups 4 }

udpEndpointProcessGroup OBJECT-GROUP
    OBJECTS    { udpEndpointProcess }
    STATUS     current
    DESCRIPTION
           "The object mapping a UDP 'endpoint' to a system process."
    ::= { udpMIBGroups 5 }


udpEndpointStatsGroup OBJECT-GROUP
    OBJECTS    { udpEndpointInDatagrams, udpEndpointOutDatagrams,
                 udpEndpointInOctets, udpEndpointOutOctets }
    STATUS     current
    DESCRIPTION
           "The group of objects providing statistics about UDP
            'endpoints'."
    ::= { udpMIBGroups 6 }

udpEndpointHCDatagramStatsGroup OBJECT-GROUP
    OBJECTS    { udpEndpointHCInDatagrams, udpEndpointHCOutDatagrams }
    STATUS     current
    DESCRIPTION
           "The group of objects to provide statitics about UDP
            'endpoints' on very high speed UDP implementations."
    ::= { udpMIBGroups 7 }

udpEndpointHCOctetStatsGroup OBJECT-GROUP
    OBJECTS    { udpEndpointHCInOctets, udpEndpointHCOutOctets }
    STATUS     current
    DESCRIPTION
           "The group of objects to provide statistics about UDP
            'endpoints' on high speed UDP implementations."
    ::= { udpMIBGroups 8 }

END
```

5. Acknowledgements

This document contains a modified subset of RFC 1213 and updates RFC **2013 and RFC 2454**.

6. Contributors

Much of Keith McCloghrie's text from RFC1213/RFC2013 remains in this document, and the structure of the MIB is due to him.

Mike Daniele wrote the original IPv6 UDP MIB in RFC2454.

Juergen Schoenwalder provided much of the text for section 3.

7. References

[1] Rose, M. and K. McCloghrie, "Management Information Base for Network Management of TCP/IP-based internets", RFC 1213, March 1991.

[2] K. McCloghrie, "SNMPv2 Management Information Base for the User Datagram Protocol using SMIv2", RFC 2013, November 1996.

[3] M. Daniele, "IP Version 6 Management Information Base for the User Datagram Protocol", RFC 2454, December 1998.

[4] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DARPA, August 1980.

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

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

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

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

[9] 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.

[10] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M., and S. Waldbusser, "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.

[11] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M., and S. Waldbusser, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.

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

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

[14] 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.

[15] Case, J., Harrington D., Presuhn R., and B. Wijnen, "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)", RFC 2572, April 1999.

[16] Blumenthal, U., and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", RFC 2574, April 1999.

[17] 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.

[18] Levi, D., Meyer, P., and B. Stewart, "SNMPv3 Applications", RFC 2573, April 1999.

[19] Wijnen, B., Presuhn, R., and K. McCloghrie, "View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)", RFC 2575, April 1999.

[20] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction to Version 3 of the Internet-standard Network Management Framework", RFC 2570, April 1999.

8. Security Considerations

There are no management objects defined in this MIB that have a MAX-ACCESS clause of read-write and/or read-create. So, if this MIB is implemented correctly, then there is no risk that an intruder can alter or create any management objects of this MIB via direct SNMP SET operations.

There are a number of managed objects in this MIB that may contain sensitive information. These are:

o The udpEndpointLocalPort and udpLocalPort objects can be used to identify what ports are open on the machine and can thus what attacks are likely to succeed, without the attacker having to run a port scanner.

It is thus important to control even GET access to these objects and possibly to even encrypt the values of these object when sending them over the network via SNMP. Not all versions of SNMP provide features for such a secure environment.

SNMPv1 by itself is not a secure environment. Even if the network itself is secure (for example by using IPSec), even then, there is no control as to who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in this MIB.

It is recommended that the implementers consider the security features as provided by the SNMPv3 framework. Specifically, the use of the User-based Security Model RFC 2574 [16] and the View-based Access Control Model RFC 2575 [19] is recommended.

It is then a customer/user responsibility to ensure that the SNMP entity giving access to an instance of this MIB, is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

9. Editor's Address

Bill Fenner
AT&T Labs -- Research
75 Willow Rd
Menlo Park, CA 94025
USA

Email: fenner@research.att.com

10. Full Copyright Statement