Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 8, 2007

A. Persson
SUN
P. Schauer
A. Durand
Comcast
D. Thaler
Microsoft

October 5, 2006

Management Information Base for TCP and UDP processes draft-persson-v6ops-mib-issue-01.txt

Status of this Memo

Copyright Notice

Abstract

In RFC 4113 and 4022 there is a set of objects that have some outstanding issues. This document provides a short discussion of the issues and how they can be addressed.

Table of Contents

1   Introduction

Between RFC 4113 and 4022 there are several objects that have unclear behavior, or limited functionality on some platforms. Some updates are needed in order to guarantee uniform behavior and functionality across all entities implementing the RFCs. Specifically, the objects in question are tcpConnectionProcess, tcpListenerProcess, udpEndpointProcess (collectively referred to as Process objects) and udpEndpointInstance (Instance object).

2 Issues

2.1 Process Objects

The Process objects are all described as the system process associated with a particular connection. If the object has a non-zero value, it is expected to correspond to a row in either HOST-RESOURCES-MIB::hrSWRunIndex or SYSAPPL-MIB::sysApplElmRunIndex. An object value of zero is used to identify cases where the connection is not associated with a processes.

One of the usages for the Process objects is to track down misbehaving applications. For example, if an administrator detects unwanted data traffic that is sent to or from a machine under his/her control, then the connection tuple could be located in either the TCP or UDP connection tables. Since each entry in the table includes the process id of the controlling application, the administrator can force the application to stop.

Establishing a one-to-one association between processes and connections works well on systems that only allow such behavior. However, on certain platforms it is possible to have multiple processes that share a single connection. An example of such behavior can be seen in most UNIX environments, where a process initially opens a new connection, and then uses the fork() system call to create one or more child processes. Each of the child processes will then have access to the connection opened by the parent process. However, it would not be possible to report multiple processes to the administrator using the current tables, which limits the functionality.

2.2 Instance Object

The second issue is udpEndpointInstance, which is part of udpEndpointTable. The table is defined in RFC 4113 and it contains all connected and listening UDP endpoints. The entries in the table are indexed using the connection tuple as well as an Instance object. The Instance is used to distinguish between multiple identical UDP endpoints, which might happen, for example, if multicast is used. The assignment of instance values is implementation specific, and to give flexibility for implementors, the description is very minimal. Specifically, the description does not state if instance values can be reused, or if the values should be allocated in any particular order. In certain situations, the lack of such information can make it hard for administrators to detect system issues.

To illustrate the issues, consider the following scenarios:

Scenario 1: Assume there is a process providing a service, and the UDP endpoint associated with the service has an identifying tuple A. Also, the system has assigned the endpoint an instance value of x, and so the endpoint's index is A.x. An administrator wants to ensure that the service is operating properly, and is doing so by looking up A.x in udpEndpointTable at a regular interval. However, the presence of A.x in udpEndpointTable does not necessarily mean that the service is running properly. It could be the case that the service is constantly restarting due to errors, and the system is reusing the instance value x.

Scenario 2: Assume there are multiple UDP endpoints that are receiving multicast packets from a specific sender. All the endpoints will therefore have the same tuple, but different instance values. However, the instance values do not give any indication of how long the different endpoints have been active. It would therefore be difficult to determine the status of the different endpoints.

3 Suggested Approaches

3.1 Process Objects

Enumerating all processes associated with connections will be done by introducing new tables. The tables are optional, and can be provided by those platforms that want to extend the functionality of RFC 4022 and 4113.

RFC 4113 and 4022 define three connection tables: tcpConnectionTable, tcpListenerTable, and udpEndpointTable, which are indexed using connection tuples (the udpEndpointTable also uses the Instance object, but we include that as part of the tuple in the following discussion). For each connection table, we define two new tables: (1) a Creation information table, and (2) a Process information table, resulting in total of six new tables.

The Creation Information tables, which are indexed using connection tuples, contains information about how and when a connection was created. More specifically, it contains the id of the process that created the connection, and when the creation event occurred. It is possible for a connection to continue, even if the creating process exits. For example, this could happen if the creating process was sharing the connection with other processes. Therefore, unlike the Process objects, the creator id does not have to correspond to a row in HOST-RESOURCES-MIB::hrSWRunIndex or SYSAPPL-MIB::sysApplElmRunIndex. The creation time can be used to determine if the id corresponds to a running process. Also, the Creation Information tables augment the existing connection tables, and therefore share the same life-time properties.

The Process tables, which are indexed using the connection tuple and the process id, are used to enumerate all active processes that are associated with connections. For each process, a corresponding row is expected to be available in either HOST-RESOURCES-MIB::hrSWRunIndex or SYSAPPL-MIB::sysApplElmRunIndex, if those tables are supported. Similarly, a connection tuple should only be present in the Process tables if there is a corresponding row in tcpConnectionTable, tcpListenerTable, or udpEndpointTable.

3.2 Instance Object

The basic description of the Instance object will remain as-is to ensure flexibility for all implementations. However, in a future update of RFC 4113, a clarification of the Instance object would be provided by adding an example to the description. One possible example would be:

"The instance value could be obtained from a counter that is incremented each time a new UDP endpoint is created. Once the counter wraps around, care must be taken to ensure that newly created indexes are unique."

The issue regarding not being able to detect change is no longer a problem, as long as the Creation Information tables are being used. Detecting whether a change has occurred can then be done by examining the creation time of the connection.

4 Process Information MIB Definitions

4.1 TCP Process Information MIB

```
TCP-PROC-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, Integer32, Unsigned32,
    Gauge32, Counter32, Counter64, IpAddress, mib-2, TimeTicks
                                        FROM SNMPv2-SMI
    MODULE-COMPLIANCE, OBJECT-GROUP    FROM SNMPv2-CONF
    InetAddress, InetAddressType,
    InetPortNumber                      FROM INET-ADDRESS-MIB
    tcpConnectionEntry, tcpListenerEntry
                                                    FROM TCP-MIB;

tcpProcMIB MODULE-IDENTITY
    LAST-UPDATED      "200610010000Z"
    ORGANIZATION        "IETF IPv6 Working Group"
    CONTACT-INFO
        "Alain Durand
        Comcast Cable
        1500 Market st
        Philadelphia
        PA 19102 USA
        Email: alain_durand@cable.comcast.com

        Anders Persson
        SUN Microsystems inc.
        17 Network Circle
        Menlo Park
        CA 94025 USA
        Email: anders.persson@sun.com

        Paul Schauer
        Comcast Cable
        183 Inverness Dr West
        Englewood
        CO 80112 USA
        Email: paul_schauer@cable.comcast.com

        David Thaler
        Microsoft
        One Microsoft Way
        Redmond
        WA 98052 USA
        Email: dthaler@microsoft.com"
```

```
    DESCRIPTION
           "Test branch for proposed TCP connection process information
            tables"

    REVISION    "200610010000Z"
    DESCRIPTION
        "Initial version"

    ::= { mib-2 990 }

tcpProc      OBJECT IDENTIFIER ::= { mib-2 992 }

--
-- The proposed new TCP Connection Information table
--

tcpConnectionInfoTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF TcpConnectionInfoEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "A table containing additional information about existing TCP
            connections. This table augments the existing
            tcpConnectionTable by providing information for the process
            that created the connection on the listed address/port,
            not just the process currently associated with the
            connection. This aids identifying processes sharing
            connections on the same port."

    ::= { tcpProc 1 }

tcpConnectionInfoEntry OBJECT-TYPE
    SYNTAX     TcpConnectionInfoEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "A conceptual row of the tcpConnectionInfoTable containing
            information about a particular current TCP connection.
            The addition of the tcpConnectionInfoCreatorPID and
            tcpConnectionInfoProcessCreateTime data provides an operator
            an explicit way to relate network connections with
            running processes."
    AUGMENTS { tcpConnectionEntry }

    ::= { tcpConnectionInfoTable 1 }

TcpConnectionInfoEntry ::= SEQUENCE {
        tcpConnectionInfoCreatorPID         Unsigned32,
```

```
        tcpConnectionInfoProcessCreateTime  TimeTicks
    }

tcpConnectionInfoCreatorPID OBJECT-TYPE
    SYNTAX     Unsigned32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The system's process ID for the process that created
            this connection, even if this process no longer exists
            or is no longer associated with this connection."

    ::= { tcpConnectionInfoEntry 1 }

tcpConnectionInfoProcessCreateTime OBJECT-TYPE
    SYNTAX     TimeTicks
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "This field provides the time the process created the
            connection on this port."

    ::= { tcpConnectionInfoEntry 2 }

--
-- The proposed new TCP Connection Process table
--

tcpConnectionProcTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF TcpConnectionProcEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "A table containing additional information about existing TCP
            connections. This table delivers functionality
            beyond the existing tcpConnectionTable
            by providing an entry for each process that is associated
            with the connection for operating systems that support this
            functionality. An entry in the tcpConnectionTable implies
            the existance of one or more entries in this table for the
            connection, and vice-versa."
    ::= { tcpProc 2 }

tcpConnectionProcEntry OBJECT-TYPE
    SYNTAX     TcpConnectionProcEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
```

```
           "A conceptual row of the tcpConnectionProcTable containing
            information about a particular current TCP connection.
            Each row of this table is transient in that it ceases to
            exist when (or soon after) the parent connection that
            created the connection exits."
    INDEX   { tcpConnectionProcLocalAddressType,
              tcpConnectionProcLocalAddress,
              tcpConnectionProcLocalPort,
              tcpConnectionProcRemAddressType,
              tcpConnectionProcRemAddress,
              tcpConnectionProcRemPort,
              tcpConnectionProcPID }
    ::= { tcpConnectionProcTable 1 }

TcpConnectionProcEntry ::= SEQUENCE {
        tcpConnectionProcLocalAddressType   InetAddressType,
        tcpConnectionProcLocalAddress       InetAddress,
        tcpConnectionProcLocalPort          InetPortNumber,
        tcpConnectionProcRemAddressType     InetAddressType,
        tcpConnectionProcRemAddress         InetAddress,
        tcpConnectionProcRemPort            InetPortNumber,
        tcpConnectionProcPID                Unsigned32
    }

tcpConnectionProcLocalAddressType OBJECT-TYPE
    SYNTAX     InetAddressType
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "The address type of tcpConnectionProcLocalAddress."
    ::= { tcpConnectionProcEntry 1 }

tcpConnectionProcLocalAddress OBJECT-TYPE
    SYNTAX     InetAddress
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "The local IP address for this TCP connection.  The type
            of this address is determined by the value of
            tcpConnectionProcLocalAddressType.
            As this object is used in the index for the
            tcpConnectionProcTable, implementors should be
            careful not to create entries that would result in OIDs
            with more than 128 subidentifiers; otherwise the information
            cannot be accessed by using SNMPv1, SNMPv2c, or SNMPv3."
    ::= { tcpConnectionProcEntry 2 }

tcpConnectionProcLocalPort OBJECT-TYPE
```

```
    SYNTAX     InetPortNumber
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "The local port number for this TCP connection."
    ::= { tcpConnectionProcEntry 3 }

tcpConnectionProcRemAddressType OBJECT-TYPE
    SYNTAX     InetAddressType
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "The address type of tcpConnectionProcRemAddress."
    ::= { tcpConnectionProcEntry 4 }

tcpConnectionProcRemAddress OBJECT-TYPE
    SYNTAX     InetAddress
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "The remote IP address for this TCP connection.  The type
            of this address is determined by the value of
            tcpConnectionInfoRemAddressType.

            As this object is used in the index for the
            tcpConnectionProcTable, implementors should be
            careful not to create entries that would result in OIDs
            with more than 128 subidentifiers; otherwise the information
            cannot be accessed by using SNMPv1, SNMPv2c, or SNMPv3."
    ::= { tcpConnectionProcEntry 5 }

tcpConnectionProcRemPort OBJECT-TYPE
    SYNTAX     InetPortNumber
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "The remote port number for this TCP connection."
    ::= { tcpConnectionProcEntry 6 }


tcpConnectionProcPID OBJECT-TYPE
    SYNTAX     Unsigned32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The system's process ID for the process sharing
            this connection. This process corresponds to a row
            in HOST-RESOURCES-MIB::hrSWRunIndex and
```

```
            SYSAPPL-MIB::sysApplElmRunIndex for operating systems
            that support this functionality and the corresponding MIBs."

    ::= { tcpConnectionProcEntry 8 }

-- The TCP Listener Information table

tcpListenerInfoTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF TcpListenerInfoEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "A table containing additional information about existing TCP
            listeners. This table augments the existing tcpListenerTable
            by providing information for the process that created the
            listener on the listed address/port, not just the
            process currently associated with the listener. This
            aids identifying multiple processes listening on the
            same port."
    ::= { tcpProc 3 }

tcpListenerInfoEntry OBJECT-TYPE
    SYNTAX     TcpListenerInfoEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "A conceptual row of the tcpListenerProcTable containing
            information about a particular TCP listener."
    AUGMENTS { tcpListenerEntry }

    ::= { tcpListenerInfoTable 1 }

TcpListenerInfoEntry ::= SEQUENCE {
        tcpListenerInfoCreatorPID             Unsigned32,
        tcpListenerInfoProcessCreateTime      TimeTicks
    }

tcpListenerInfoCreatorPID OBJECT-TYPE
    SYNTAX     Unsigned32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The system's process ID for the process that created
            this listener, even if this process no longer exists
            or is no longer associated with this connection."
    ::= { tcpListenerInfoEntry 1 }

tcpListenerInfoProcessCreateTime OBJECT-TYPE
```

```
    SYNTAX     TimeTicks
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "This field provides the time the process started
            listening on this port."
    ::= { tcpListenerInfoEntry 2 }

-- The TCP Listener Process table

tcpListenerProcTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF TcpListenerProcEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "A table containing additional information about existing
            TCP listeners. This table delivers functionality beyond
            the existing tcpListenerTable by providing an entry
            for each process that is associated with the listener
            for operating systems that support this functionality.
            An entry in the tcpListenerTable implies the existance of
            one or more entries in this table for the listener, and
            vice-versa.  A listening application can be represented
            in three possible ways:

            1. An application that is willing to accept both IPv4 and
               IPv6 datagrams is represented by
               a tcpListenerProcLocalAddressType of unknown (0) and
               a tcpListenerProcLocalAddress of ''h (a zero-length
               octet-string).

            2. An application that is willing to accept only IPv4 or
               IPv6 datagrams is represented by a
               tcpListenerProcLocalAddressType of the appropriate
               address type and a tcpListenerProcLocalAddress of
               '0.0.0.0' or '::' respectively.

            3. An application that is listening for data destined
               only to a specific IP address, but from any remote
               system, is represented by a
               tcpListenerProcLocalAddressType of an appropriate
               address type, with tcpListenerProcLocalAddress
               as the specific local address.

            NOTE: 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))."
    ::= { tcpProc 4 }

tcpListenerProcEntry OBJECT-TYPE
    SYNTAX     TcpListenerProcEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "A conceptual row of the tcpListenerProcTable containing
            information about a particular TCP listener."
    INDEX   { tcpListenerProcLocalAddressType,
              tcpListenerProcLocalAddress,
              tcpListenerProcLocalPort,
              tcpListenerProcPID }
    ::= { tcpListenerProcTable 1 }

TcpListenerProcEntry ::= SEQUENCE {
        tcpListenerProcLocalAddressType       InetAddressType,
        tcpListenerProcLocalAddress           InetAddress,
        tcpListenerProcLocalPort              InetPortNumber,
        tcpListenerProcPID                    Unsigned32
    }

tcpListenerProcLocalAddressType OBJECT-TYPE
    SYNTAX     InetAddressType
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "The address type of tcpListenerProcLocalAddress.  The value
            should be unknown (0) if connection initiations to all
            local IP addresses are accepted."
    ::= { tcpListenerProcEntry 1 }

tcpListenerProcLocalAddress OBJECT-TYPE
    SYNTAX     InetAddress
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "The local IP address for this TCP connection.
            The value of this object can be represented in three
            possible ways, depending on the characteristics of the
            listening application:

            1. For an application willing to accept both IPv4 and
               IPv6 datagrams, the value of this object must be
               ''h (a zero-length octet-string), with the value
               of the corresponding tcpListenerProcLocalAddressType
```

```
               object being unknown (0).

            2. For an application willing to accept only IPv4 or
               IPv6 datagrams, the value of this object must be
               '0.0.0.0' or '::' respectively, with
               tcpListenerProcLocalAddressType representing the
               appropriate address type.

            3. For an application which is listening for data
               destined only to a specific IP address, the value
               of this object is the specific local address, with
               tcpListenerProcLocalAddressType representing the
               appropriate address type.

            As this object is used in the index for the
            tcpListenerProcTable, implementors should be
            careful not to create entries that would result in OIDs
            with more than 128 subidentifiers; otherwise the information
            cannot be accessed, using SNMPv1, SNMPv2c, or SNMPv3."
    ::= { tcpListenerProcEntry 2 }

tcpListenerProcLocalPort OBJECT-TYPE
    SYNTAX     InetPortNumber
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "The local port number for this TCP connection."
    ::= { tcpListenerProcEntry 3 }

tcpListenerProcPID OBJECT-TYPE
    SYNTAX     Unsigned32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The system's process ID for the process associated with
            this listener."
    ::= { tcpListenerProcEntry 4 }

-- compliance statements
tcpProcMIBConformance OBJECT IDENTIFIER ::= { tcpProcMIB 1 }

tcpProcMIBCompliances OBJECT IDENTIFIER ::= { tcpProcMIBConformance 1 }
tcpProcMIBGroup       OBJECT IDENTIFIER ::= { tcpProcMIBConformance 2 }

tcpProcMIBConnectionCompliance  MODULE-COMPLIANCE
    STATUS      current
    DESCRIPTION
            "The compliance statement for systems that implement the
```

```
             TCP process MIB."
    MODULE -- this module
    MANDATORY-GROUPS { tcpProcInfoGroup }
    GROUP  tcpProcProcessGroup
    DESCRIPTION
           "This group should be implemented for operating systems that
            support multiple processes sharing a single connection. It
            is left as optional to accommodate operating systems that do
            not provide sufficient information to express this data."

    ::= { tcpProcMIBCompliances 1 }

tcpProcMIBListenerCompliance    MODULE-COMPLIANCE
    STATUS      current
    DESCRIPTION
           "The compliance statement for systems that implement the
            TCP process MIB."
    MODULE -- this module
    MANDATORY-GROUPS { tcpProcListenerInfoGroup }
    GROUP  tcpProcListenerProcessGroup
    DESCRIPTION
           "This group should be implemented for operating systems that
            support multiple processes sharing a single listener. It is
            left as optional to accommodate operating systems that do
            not provide sufficient information to express this data."

    ::= { tcpProcMIBCompliances 2 }

-- units of conformance

tcpProcInfoGroup OBJECT-GROUP
    OBJECTS     { tcpConnectionInfoCreatorPID,
                  tcpConnectionInfoProcessCreateTime }
    STATUS      current
    DESCRIPTION
           "The tcpProcInfoGroup providing basic information about
            processes associated with a specific connection"

    ::= { tcpProcMIBGroups 1 }

tcpProcProcessGroup OBJECT-GROUP
    OBJECTS     { tcpConnectionProcPID }
    STATUS      current
    DESCRIPTION
           "The tcpProcProcessGroup providing specific process
            information about processes associated with a specific
            connection."
```

```
    ::= { tcpProcMIBGroups 2 }

tcpProcListenerInfoGroup OBJECT-GROUP
    OBJECTS { tcpListenerInfoCreatorPID,
              tcpListenerInfoProcessCreateTime }
    STATUS      current
    DESCRIPTION
           "The tcpProcListenerInfoGroup providing basic information
            about processes associated with a specific listener."

    ::= { tcpProcMIBGroups 3 }

tcpProcListenerProcessGroup OBJECT-GROUP
    OBJECTS { tcpListenerProcPID }
    STATUS      current
    DESCRIPTION
            "The tcpProcListenerProcessGroup providing specific process
             information about processes associated with a specific
             listener."

    ::= { tcpProcMIBGroups 4 }
END
```

4.2 UDP Process Information MIB

```
UDP-PROC-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE,
    Integer32, Counter32, Counter64,
    TimeTicks, Unsigned32,IpAddress,
    mib-2                             FROM SNMPv2-SMI

    MODULE-COMPLIANCE, OBJECT-GROUP   FROM SNMPv2-CONF
    InetAddress, InetAddressType,
    InetPortNumber                    FROM INET-ADDRESS-MIB
    udpEndpointEntry
                                      FROM UDP-MIB;

udpProcMIB MODULE-IDENTITY
   LAST-UPDATED      "200610010000Z"
   ORGANIZATION        "IETF IPv6 Working Group"
   CONTACT-INFO
        "Alain Durand
        Comcast Cable
        1500 Market st
        Philadelphia
```

```
        PA 19102 USA
        Email: alain_durand@cable.comcast.com

        Anders Persson
        SUN Microsystems inc.
        17 Network Circle
        Menlo Park
        CA 94025 USA
        Email: anders.persson@sun.com

        Paul Schauer
        Comcast Cable
        183 Inverness Dr West
        Englewood
        CO 80112 USA
        Email: paul_schauer@cable.comcast.com

        David Thaler
        Microsoft
        One Microsoft Way
        Redmond
        WA 98052 USA
        Email: dthaler@microsoft.com"

    DESCRIPTION
           "Test branch for proposed UDP listener information tables"

    REVISION    "200610010000Z"
    DESCRIPTION
        "Initial version"


    ::= { mib-2 994 }

udpProc      OBJECT IDENTIFIER ::= { mib-2 996 }

--
-- The proposed new UDP Endpoint Info table.
--

udpEndpointInfoTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF UdpEndpointInfoEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "A table containing additional information about existing UDP
            endpoints. This table augments the existing udpEndpointTable
            by providing information for the process that created the
```

```
            endpoint on the listed address/port, not just the
            process currently associated with the endpoint. This
            aids identifying processes sharing connections on the same
            port."

    ::= { udpProc 1 }

udpEndpointInfoEntry OBJECT-TYPE
    SYNTAX     UdpEndpointInfoEntry
    MAX-ACCESS not-accessible
    STATUS     current

    DESCRIPTION
        "The additional time field allows an operator to identify
        when a partcular UDP endpoint came into existance."

    AUGMENTS { udpEndpointEntry }

    ::= { udpEndpointInfoTable 1 }

UdpEndpointInfoEntry ::= SEQUENCE {
        udpEndpointInfoCreatorPID               Unsigned32,
        udpEndpointInfoProcessCreateTime        TimeTicks
        }

udpEndpointInfoCreatorPID OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The system's process ID for the process that created
      this endpoint, even if this process no longer exists
        or is no longer associated with this connection."

    ::= { udpEndpointInfoEntry 1 }

udpEndpointInfoProcessCreateTime OBJECT-TYPE
    SYNTAX      TimeTicks
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This field provides the time the process created the
        endpoint on this port.
        "
    ::= { udpEndpointInfoEntry 2 }

--
-- The proposed new UDP Endpoint process table.
```

```
--

udpEndpointProcTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF UdpEndpointProcEntry
    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.
        This table delivers functionality beyond the existing
        udpEndpointTable by providing an entry for each process that
        creates a shared endpoint on the same port for operating systems
        that support this functionality. An entry in the
        udpEndpointTable implies the existance of one or more entries in
        this table for the connection, and vice-versa."

    ::= { udpProc 2 }

udpEndpointProcEntry OBJECT-TYPE
    SYNTAX     UdpEndpointProcEntry
    MAX-ACCESS not-accessible
    STATUS     current

    DESCRIPTION
           "Information about a particular current UDP endpoint.

            Implementers need to be aware that if the total number
            of elements (octets or sub-identifiers) in
            udpEndpointProcLocalAddress and udpEndpointProcRemoteAddress
            exceeds 111, then OIDs of column instances in this table
            will have more than 128 sub-identifiers and cannot be
            accessed using SNMPv1, SNMPv2c, or SNMPv3."
    INDEX   { udpEndpointProcLocalAddressType,
              udpEndpointProcLocalAddress,
              udpEndpointProcLocalPort,
              udpEndpointProcRemoteAddressType,
              udpEndpointProcRemoteAddress,
              udpEndpointProcRemotePort,
              udpEndpointProcInstance,
              udpEndpointProcPID
             }
    ::= { udpEndpointProcTable 1 }

UdpEndpointProcEntry ::= SEQUENCE {
        udpEndpointProcLocalAddressType   InetAddressType,
        udpEndpointProcLocalAddress       InetAddress,
        udpEndpointProcLocalPort          InetPortNumber,
```

```
        udpEndpointProcRemoteAddressType  InetAddressType,
        udpEndpointProcRemoteAddress      InetAddress,
        udpEndpointProcRemotePort         InetPortNumber,
        udpEndpointProcInstance           Unsigned32,
          udpEndpointProcPID              Unsigned32
    }

udpEndpointProcLocalAddressType OBJECT-TYPE
    SYNTAX     InetAddressType
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "The address type of udpEndpointProcLocalAddress.  Only
            IPv4, IPv4z, IPv6, and IPv6z addresses are expected, or
            unknown(0) if datagrams for all local IP addresses are
            accepted."

    ::= { udpEndpointProcEntry 1 }

udpEndpointProcLocalAddress OBJECT-TYPE
    SYNTAX     InetAddress
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "The local IP address for this UDP endpoint.

            The value of this object can be represented in three
            possible ways, depending on the characteristics of the
            listening application:

            1. For an application that is willing to accept both
               IPv4 and IPv6 datagrams, the value of this object
               must be ''h (a zero-length octet-string), with
               the value of the corresponding instance of the
               udpEndpointLocalAddressType object being unknown(0).

            2. For an application that is willing to accept only IPv4
               or only IPv6 datagrams, the value of this object
               must be '0.0.0.0' or '::', respectively, while the
               corresponding instance of the
               udpEndpointLocalAddressType object represents the
               appropriate address type.

            3. For an application that is listening for data
               destined only to a specific IP address, the value
               of this object is the specific IP address for which
               this node is receiving packets, with the
               corresponding instance of the
```

```
               udpEndpointLocalAddressType object representing the
               appropriate address type.

            As this object is used in the index for the
            udpEndpointProcTable, implementors of this table should be
            careful not to create entries that would result in OIDs
            with more than 128 subidentifiers; else the information
            cannot be accessed using SNMPv1, SNMPv2c, or SNMPv3."

    ::= { udpEndpointProcEntry 2 }

udpEndpointProcLocalPort OBJECT-TYPE
    SYNTAX     InetPortNumber
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "The local port number for this UDP endpoint."

    ::= { udpEndpointProcEntry 3 }

udpEndpointProcRemoteAddressType OBJECT-TYPE
    SYNTAX     InetAddressType
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "The address type of udpEndpointProcRemoteAddress.  Only
            IPv4, IPv4z, IPv6, and IPv6z addresses are expected, or
            unknown(0) if datagrams for all remote IP addresses are
            accepted.  Also, note that some combinations of
            udpEndpointProcLocalAddressType and
            udpEndpointProcRemoteAddressType are not supported.  In
            particular, if the value of this object is not
            unknown(0), it is expected to always refer to the
            same IP version as udpEndpointProcLocalAddressType."

    ::= { udpEndpointProcEntry 4 }

udpEndpointProcRemoteAddress OBJECT-TYPE
    SYNTAX     InetAddress
    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
            udpEndpointProcRemoteAddressType and is the address of the
            remote system from which datagrams are to be accepted
```

```
            (or to which all datagrams will be sent).

            As this object is used in the index for the
            udpEndpointProcTable, implementors of this table should be
            careful not to create entries that would result in OIDs
            with more than 128 subidentifiers; else the information
            cannot be accessed using SNMPv1, SNMPv2c, or SNMPv3."

    ::= { udpEndpointProcEntry 5 }

udpEndpointProcRemotePort 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."

    ::= { udpEndpointProcEntry 6 }

udpEndpointProcInstance 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 among multiple processes 'connected' to
            the same UDP endpoint.  For example, on a system
            implementing the BSD sockets interface, this would be
            used to support the SO_REUSEADDR and SO_REUSEPORT
            socket options."

    ::= { udpEndpointProcEntry 7 }

udpEndpointProcPID OBJECT-TYPE
    SYNTAX     Unsigned32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The system's process ID for the process associated with
            this endpoint.
            This value corresponds to a row in
            HOST-RESOURCES-MIB::hrSWRunIndex and SYSAPPL-MIB::
            sysApplElmtRunIndex for operating systems that
            support this functionality and the corresponding MIBs."

    ::= { udpEndpointProcEntry 8 }
```

```
-- compliance statements
udpProcMIBConformance OBJECT IDENTIFIER ::= { udpProcMIB 1 }

udpProcMIBCompliances OBJECT IDENTIFIER ::= { udpProcMIBConformance 1 }
udpProcMIBGroup       OBJECT IDENTIFIER ::= { udpProcMIBConformance 2 }

udpProcMIBCompliance    MODULE-COMPLIANCE
    STATUS      current
    DESCRIPTION
           "The compliance statement for systems that implement the
            UDP Process MIB."
    MODULE -- this module
    MANDATORY-GROUPS { udpEndpointInfoGroup }
    GROUP  udpEndpointProcessGroup
    DESCRIPTION
           "This group should be implemented for operating systems that
            support multiple listening processes sharing a single
            address/port. It is left as optional to accommodate
            operating systems that do not provide sufficient information
            to express this data."

    ::= { udpProcMIBCompliances 1 }

-- units of conformance

udpEndpointInfoGroup OBJECT-GROUP
    OBJECTS { udpEndpointInfoCreatorPID,
              udpEndpointInfoProcessCreateTime }
    STATUS      current
    DESCRIPTION
           ""
    ::= { udpProcMIBGroups 1 }

udpEndpointProcessGroup OBJECT-GROUP
    OBJECTS { udpEndpointProcPID }
    STATUS      current
    DESCRIPTION
           ""
    ::= { udpProcMIBGroups 2 }

END
```

5 Security Considerations

The security considerations discussed in RFC 4113 and RFC 4022 apply here.

Authors' Addresses

Anders Persson
SUN Microsystems Inc.
17 Network Circle
Menlo Park, CA 94025
USA

Email: anders.persson@sun.com

Paul Schauer
Comcast
183 Inverness Dr West
Englewood, CO 80112
USA

Email: Paul_Schauer@cable.comcast.com

Alain Durand
Comcast
1500 Market St
Philadelphia, PA 19102
USA

Email: Alain_Durand@cable.comcast.com

Dave Thaler
Microsoft
One Microsoft Way
Redmond, WA 98052
USA

Email: dthaler@microsoft.com

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).