

```

--
-- file: vossysmgr.mib
--
-- COMTEK Services, Inc. NM*SysMgr MIB for VOS (Stratus, IBM System 88)
-- Release      1.5.8
-- Date         Feb, 1997
-- Author       SH
--
-- Copyright 1997-2000 COMTEK Services, Inc. All Rights Reserved.
--
-- This COMTEK Services SNMP Management Information Base Specification
-- (Specification) embodies COMTEK Services' confidential and
-- proprietary intellectual property. COMTEK Services retains all
-- title and ownership in the Specification, including any
-- revisions.
--
-- This Specification is supplied "AS IS," and COMTEK Services makes
-- no warranty, either express or implied, as to the use,
-- operation, condition, or performance of the Specification.

-- The operation of the NM*SysMgr agent for VOS is controlled via
-- the configuration file (snmpd.config), the community file (snmpd.comm),
-- the trap destination file (snmpd.trap_comm), and the user trap
-- destination file (snmpd.user_trap). See the User's Guide for a
-- complete description of each of these files.

-- VOS times are put into time ticks. CPU utilization statistics are
-- normalized by number of processors so they will fall in the range of
-- 0-100.

COMTEK-MIB DEFINITIONS ::= BEGIN

IMPORTS
    DisplayString                FROM RFC1213-MIB
    OBJECT-TYPE, TimeTicks, Counter, enterprises
                                FROM RFC1155-SMI
    comtek, comtekvosMib, comtekvosAgent
                                FROM COMTEK-DEFINITIONS-MIB;

-- The following is used to identify versions of the VOS agent
comtekVosAgent OBJECT IDENTIFIER ::= { comtekvosAgent 1 }

-- This group lists system resources available (OS, CPU, disk)
resources OBJECT IDENTIFIER      ::= { comtekvosMib 1 }

-- This group contains a table of VOS processes and statistics
processes OBJECT IDENTIFIER      ::= { comtekvosMib 2 }

-- This group contains a table of user trap data
userinfo OBJECT IDENTIFIER       ::= { comtekvosMib 3 }

vos OBJECT IDENTIFIER            ::= { resources 1 }
cpu OBJECT IDENTIFIER            ::= { resources 2 }
disk OBJECT IDENTIFIER           ::= { resources 3 }

```

```
-- The VOS agent may be configured to process data for one, selected,  
-- or all modules. By default, the agent gathers data for all accessible  
-- modules. To limit data gathering to the module on which the agent is  
-- run, specify "localModule=1" in the snmpd.config configuration file. To  
-- select a subset of modules that the agent is to monitor, specify each  
-- module to be monitored using the "monitor=<module name>" configuration  
-- parameter.
```

```
vosNumModules OBJECT-TYPE
```

```
SYNTAX INTEGER  
ACCESS read-only  
STATUS mandatory  
DESCRIPTION  
    "Number of modules."  
 ::= { vos 1 }
```

```
vosTable OBJECT-TYPE
```

```
SYNTAX SEQUENCE OF VosTableEntry  
ACCESS not-accessible  
STATUS mandatory  
DESCRIPTION  
    "A simplified version of what is read from  
    s$get_module_usage()."  
 ::= { vos 2 }
```

```
vosTableEntry OBJECT-TYPE
```

```
SYNTAX VosTableEntry  
ACCESS not-accessible  
STATUS mandatory  
DESCRIPTION  
    "A row in the module table."  
INDEX { vosModuleNumber }  
 ::= { vosTable 1 }
```

```
VosTableEntry ::= SEQUENCE {  
    vosModuleNumber INTEGER,  
    vosModuleName DisplayString,  
    vosVersion DisplayString,  
    vosCPUtype DisplayString,  
    vosNumCPUs INTEGER,  
    vosModPaging INTEGER  
}
```

```
vosModuleNumber OBJECT-TYPE
```

```
SYNTAX INTEGER  
ACCESS read-only  
STATUS mandatory  
DESCRIPTION  
    "Table index number assigned to this module."  
 ::= { vosTableEntry 1 }
```

```
vosModuleName OBJECT-TYPE
```

```
SYNTAX DisplayString (SIZE (0..66))  
ACCESS read-only  
STATUS mandatory  
DESCRIPTION
```

```

        "Name of this module."
        ::= { vosTableEntry 2 }

vosVersion OBJECT-TYPE
    SYNTAX DisplayString (SIZE (0..100))
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The version numbers for this copy of VOS."
        ::= { vosTableEntry 3 }

vosCPUtype OBJECT-TYPE
    SYNTAX DisplayString (SIZE (0..100))
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The type and model number of the CPU(s)."
        ::= { vosTableEntry 4 }

vosNumCPUs OBJECT-TYPE
    SYNTAX INTEGER (1..32)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Number of CPU(s) currently running."
        ::= { vosTableEntry 5 }

vosModPaging OBJECT-TYPE
    SYNTAX INTEGER (0..100)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Percent of paging space used for module."
        ::= { vosTableEntry 6 }

-- The red light table is a circular buffer of hardware log messages from
-- monitored modules.  This table is not organized by module, rather the
-- messages are placed in the table in the order in which they arrive.
-- This table is being implemented as a circular buffer to avoid the
-- problem of the user who enables these messages and never cleans the
-- table.  There is currently a limit of 100 log items in this table.

vosNumRedLights OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Number of entries in the red light table."
        ::= { vos 3 }

vosRedLightTable OBJECT-TYPE
    SYNTAX SEQUENCE OF VosRedLightTableEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A table of messages from the >system>hardware_log.*."
        ::= { vos 4 }

```

```

vosRedLightTableEntry OBJECT-TYPE
    SYNTAX VosRedLightTableEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A row in the red light table."
    INDEX { vosRedLightNumber }
    ::= { vosRedLightTable 1 }

VosRedLightTableEntry ::= SEQUENCE {
    vosRedLightNumber      INTEGER,
    vosRedLightModuleName DisplayString,
    vosRedLightMessage     DisplayString
}

vosRedLightNumber OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Table index number assigned to this red light entry."
    ::= { vosRedLightTableEntry 1 }

vosRedLightModuleName OBJECT-TYPE
    SYNTAX DisplayString (SIZE (0..66))
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Name of module event occurred on."
    ::= { vosRedLightTableEntry 2 }

vosRedLightMessage OBJECT-TYPE
    SYNTAX DisplayString (SIZE (0..112))
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Message from hardware log."
    ::= { vosRedLightTableEntry 3 }

vosConfigFileName OBJECT-TYPE
    SYNTAX DisplayString (SIZE (0..256))
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "Full path name of current configuration file used by agent.
        The agent always uses the file snmpd.config as its configuration
        file on startup. Set requests on this variable have not been
        implemented. To modify configuration file parameters, make
        the necessary changes to the snmpd.config file and stop and
        restart the agent."
    ::= { vos 5 }

vosInputQueueMessage OBJECT-TYPE
    SYNTAX OCTET STRING (SIZE (0..512))
    ACCESS read-write
    STATUS mandatory

```

DESCRIPTION

"Most recent remote console message sent by NMS to agent. This MIB item is the method in which remote console commands are entered by the NMS. Performing a set-request on this MIB variable creates a new remote console command. The agent places each new remote console command in the Input Queue named below for processing by the NM\*Console process."

::= { vos 6 }

vosInputQueueName OBJECT-TYPE

SYNTAX DisplayString (SIZE (0..256))

ACCESS read-only

STATUS mandatory

DESCRIPTION

"Input Queue name where the agent is to write remote console messages for processing by NM\*Console. This name is specified in the configuration file and must be identical to the name of the queue specified when starting the NM\*Console process."

::= { vos 7 }

vosTrapCommFileName OBJECT-TYPE

SYNTAX DisplayString (SIZE (0..256))

ACCESS read-write

STATUS mandatory

DESCRIPTION

"Full path name of the current trap destination file being used by the agent. The agent always starts up using the file snmpd.trap\_comm. Set-requests on this file name cause the agent to reinitialize its trap destinations using the data in the new file name."

::= { vos 8 }

vosUserTrapCommFileName OBJECT-TYPE

SYNTAX DisplayString (SIZE (0..256))

ACCESS read-write

STATUS mandatory

DESCRIPTION

"Full path name of the current user trap destination file being used by the agent. If the file snmpd.user\_trap exists when the agent starts up, user traps are routed using this file. If this file is not found when the agent starts up, the file snmpd.trap\_comm will be used to specify user trap destinations. Set-requests on this file name cause the agent to reinitialize its user trap destinations using the data in the new file name."

::= { vos 9 }

cpuModLoad OBJECT IDENTIFIER ::= { cpu 1 }

cpuModUsage OBJECT IDENTIFIER ::= { cpu 2 }

-- The cpuModLoadTable collects load statistics by module.

cpuModLoadTable OBJECT-TYPE

SYNTAX SEQUENCE OF CpuModLoadEntry

ACCESS not-accessible

STATUS mandatory

```

DESCRIPTION
    "CPU load by module."
    ::= { cpuModLoad 1 }

cpuModLoadEntry OBJECT-TYPE
    SYNTAX CpuModLoadEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "CPU load table structure."
    INDEX { cpuModLoadIndex }
    ::= { cpuModLoadTable 1 }

CpuModLoadEntry ::= SEQUENCE {
    cpuModLoadIndex          INTEGER,
    cpuModLoadName           DisplayString,
    cpuModLoadOneMinute      INTEGER,
    cpuModLoadFiveMinute     INTEGER,
    cpuModLoadFifteenMinute  INTEGER,
    cpuModLoadIntsOneMinute  INTEGER,
    cpuModLoadIntsFiveMinute INTEGER,
    cpuModLoadIntsFifteenMinute INTEGER,
    cpuModLoadPFsOneMinute   INTEGER,
    cpuModLoadPFsFiveMinute  INTEGER,
    cpuModLoadPFsFifteenMinute INTEGER
}

cpuModLoadIndex OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Location in table."
    ::= { cpuModLoadEntry 1 }

cpuModLoadName OBJECT-TYPE
    SYNTAX DisplayString (SIZE (0..66))
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Name of module."
    ::= { cpuModLoadEntry 2 }

cpuModLoadOneMinute OBJECT-TYPE
    SYNTAX TimeTicks
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "CPU seconds used in the previous minute by module."
    ::= { cpuModLoadEntry 3 }

cpuModLoadFiveMinute OBJECT-TYPE
    SYNTAX TimeTicks
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "CPU seconds used in previous five minutes by module."

```

```

 ::= { cpuModLoadEntry 4 }

cpuModLoadFifteenMinute OBJECT-TYPE
    SYNTAX TimeTicks
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "CPU seconds used in previous fifteen minutes by module."
    ::= { cpuModLoadEntry 5 }

cpuModLoadIntsOneMinute OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Number of interrupts in one minute by module."
    ::= { cpuModLoadEntry 6 }

cpuModLoadIntsFiveMinute OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Number of interrupts in five minutes by module."
    ::= { cpuModLoadEntry 7 }

cpuModLoadIntsFifteenMinute OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Number of interrupts in fifteen minutes by module."
    ::= { cpuModLoadEntry 8 }

cpuModLoadPFsOneMinute OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Number of page faults (user, system, and server) in one minute
        by module."
    ::= { cpuModLoadEntry 9 }

cpuModLoadPFsFiveMinute OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Number of page faults (user, system and server) in five minutes
        by module."
    ::= { cpuModLoadEntry 10 }

cpuModLoadPFsFifteenMinute OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION

```

```

        "Number of page faults (user, system and server) in fifteen minutes
        by module."
    ::= { cpuModLoadEntry 11 }

-- The cpuModUsageTable shows the amount of CPU usage in percent. This is
-- normalized by the number of processors so that the range is 0-100. The
-- data in this table can be used to generate traps whenever overall CPU
-- usage or interrupt CPU usage reaches or exceeds user set thresholds.
-- The variables CpuUtil and IntUtil in the configuration file may be
-- used to specify the threshold for excessive CPU usage traps
-- (cpuUsageExcessive) and excessive CPU usage by interrupt traps
-- (interruptCpuUsageExcessive), respectively.

cpuModUsageTable OBJECT-TYPE
    SYNTAX SEQUENCE OF CpuModUsageEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "CPU usage by module."
    ::= { cpuModUsage 1 }

cpuModUsageEntry OBJECT-TYPE
    SYNTAX CpuModUsageEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "CPU usage table structure."
    INDEX { cpuModUsageIndex }
    ::= { cpuModUsageTable 1 }

CpuModUsageEntry ::= SEQUENCE {
    cpuModUsageIndex      INTEGER,
    cpuModUsageName       DisplayString,
    cpuModUsageUser       INTEGER,
    cpuModUsageSystem     INTEGER,
    cpuModUsageInterrupts INTEGER,
    cpuModUsageServer     INTEGER,
    cpuModUsageIdle       INTEGER
}

cpuModUsageIndex OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Location in table."
    ::= { cpuModUsageEntry 1 }

cpuModUsageName OBJECT-TYPE
    SYNTAX DisplayString (SIZE (0..66))
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Name of module."
    ::= { cpuModUsageEntry 2 }

cpuModUsageUser OBJECT-TYPE

```

SYNTAX INTEGER (0..100)  
ACCESS read-only  
STATUS mandatory  
DESCRIPTION  
    "Percent of CPU utilized by user processes on this module  
    during the previous minute."  
::= { cpuModUsageEntry 3 }

cpuModUsageSystem OBJECT-TYPE  
SYNTAX INTEGER (0..100)  
ACCESS read-only  
STATUS mandatory  
DESCRIPTION  
    "Percent of CPU utilized by system processes on this module  
    during the previous minute."  
::= { cpuModUsageEntry 4 }

cpuModUsageInterrupts OBJECT-TYPE  
SYNTAX INTEGER (0..100)  
ACCESS read-only  
STATUS mandatory  
DESCRIPTION  
    "Percent of CPU utilized by interrupts on this module during  
    the previous minute."  
::= { cpuModUsageEntry 5 }

cpuModUsageServer OBJECT-TYPE  
SYNTAX INTEGER (0..100)  
ACCESS read-only  
STATUS mandatory  
DESCRIPTION  
    "Percent of CPU utilized by network server processes on this  
    module during the previous minute."  
::= { cpuModUsageEntry 6 }

cpuModUsageIdle OBJECT-TYPE  
SYNTAX INTEGER (0..100)  
ACCESS read-only  
STATUS mandatory  
DESCRIPTION  
    "Percent of time the CPUs were idle while no processes were  
    ready to run during the previous minute."  
::= { cpuModUsageEntry 7 }

-- The disk system statistics are stored in a table, one row per disk. This  
-- is for all monitored modules. Traps are sent when disk utilization  
-- by the file partition or page partition reaches or exceeds limits set by  
-- the agent configuration files. The "diskSystemNumberEntries" should be  
-- read first to avoid querying nonexistent rows in the table. The  
-- configuration file variables DiskUse, MinFreeRecords, PageUse, and  
-- DiskAlarm are used to control trap message generation based on disk  
-- information. DiskUse specifies the file partition threshold at which  
-- diskFull traps are sent, MinFreeRecords specifies the threshold for  
-- the diskNoFreeSpace trap, and PageUse specifies the paging file  
-- partition at which pageFull traps are sent. The DiskAlarm parameter  
-- specifies how frequently these trap messages should be repeated.

```
diskSystemNumberEntries OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "Number of entries in the disk system table."
    ::= { disk 1 }
```

```
diskSystemTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF DiskSystemEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
        "Table of disk system statistics."
    ::= { disk 2 }
```

```
diskSystemEntry OBJECT-TYPE
    SYNTAX  DiskSystemEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
        "Disk system table structure."
    INDEX { diskSystemIndex }
    ::= { diskSystemTable 1 }
```

```
DiskSystemEntry ::= SEQUENCE {
    diskSystemIndex      INTEGER,
    diskModuleName      DisplayString,
    diskSystemName      DisplayString,
    diskSystemType      DisplayString,
    diskSystemSize      INTEGER,
    diskPartitionSize   INTEGER,
    diskPartitionUsed   INTEGER,
    diskPagePartitionSize  INTEGER,
    diskPagePartitionUsed  INTEGER,
    diskSystemFatalErrors Counter,
    diskSystemDataErrors Counter,
    diskSystemDiskReads Counter,
    diskSystemDiskWrites Counter,
    diskPartitionFree   INTEGER
}
```

```
diskSystemIndex OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "Location in table."
    ::= { diskSystemEntry 1 }
```

```
diskModuleName OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..66))
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "Name of module."
    ::= { diskSystemEntry 2 }
```

```

diskSystemName OBJECT-TYPE
    SYNTAX DisplayString (SIZE (0..66))
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Name of disk for module."
    ::= { diskSystemEntry 3 }

diskSystemType OBJECT-TYPE
    SYNTAX DisplayString (SIZE (0..66))
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Type of disk: Logical, Physical or Duplexed."
    ::= { diskSystemEntry 4 }

diskSystemSize OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Size of disk in 4096 byte blocks."
    ::= { diskSystemEntry 5 }

diskPartitionSize OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Size of file partition in 4096 byte blocks."
    ::= { diskSystemEntry 6 }

diskPartitionUsed OBJECT-TYPE
    SYNTAX INTEGER (0..100)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Percent of disk partition used."
    ::= { diskSystemEntry 7 }

diskPagePartitionSize OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Size of page partition in 4096 byte blocks."
    ::= { diskSystemEntry 8 }

diskPagePartitionUsed OBJECT-TYPE
    SYNTAX INTEGER (0..100)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Percent of page partition used."
    ::= { diskSystemEntry 9 }

```

```

diskSystemFatalErrors OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Number of fatal errors for disk."
    ::= { diskSystemEntry 10 }

diskSystemDataErrors OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Number of data errors for disk."
    ::= { diskSystemEntry 11 }

diskSystemDiskReads OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The total number of disk reads since the module was loaded."
    ::= { diskSystemEntry 12 }

diskSystemDiskWrites OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The total number of disk writes since the module was loaded."
    ::= { diskSystemEntry 13 }

--NEW in version 1.5.8
diskPartitionFree OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Number of unused 4096 byte blocks in partition."
    ::= { diskSystemEntry 14 }

-- The process statistics are stored in a table, one row per process. This
-- is for all monitored modules. The "psNumProcs" should be read first
-- to avoid querying nonexistent rows in the table. Since this table is
-- indexed on the process identifier assigned by VOS when the process
-- starts up, this is a sparse table.

psNumProcs OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Number of processes."
    ::= { processes 1 }

psTable OBJECT-TYPE
    SYNTAX SEQUENCE OF PsTableEntry

```

```

ACCESS not-accessible
STATUS mandatory
DESCRIPTION
    "A simplified version of what is read from
    s$get_processes_info()."
::= { processes 2 }

psTableEntry OBJECT-TYPE
SYNTAX PsTableEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION
    "A row in the process table."
INDEX { psProcessId }
::= { psTable 1 }

PsTableEntry ::= SEQUENCE {
    psProcessId          INTEGER,
    psModuleName         DisplayString,
    psName               DisplayString,
    psPersonName        DisplayString,
    psGroupName         DisplayString,
    psProgramName       DisplayString,
    psInvokingProcessId INTEGER,
    psState              INTEGER,
    psPriority           INTEGER,
    psCpuTime           TimeTicks,
    psPageFaultTime    TimeTicks,
    psPageFaults        Counter,
    psWorkingSetSize    INTEGER,
    psDiskReads         Counter,
    psDiskWrites        Counter,
    psNumInstances      INTEGER,
    psNumProgramInstances INTEGER
}

psProcessId OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "VOS process identifier."
::= { psTableEntry 1 }

psModuleName OBJECT-TYPE
SYNTAX DisplayString (SIZE (0..66))
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "Name of module."
::= { psTableEntry 2 }

psName OBJECT-TYPE
SYNTAX DisplayString (SIZE (0..32))
ACCESS read-only
STATUS mandatory
DESCRIPTION

```

```

        "Process name."
 ::= { psTableEntry 3 }

psPersonName OBJECT-TYPE
    SYNTAX DisplayString (SIZE (0..32))
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Name of user who started the process."
 ::= { psTableEntry 4 }

psGroupName OBJECT-TYPE
    SYNTAX DisplayString (SIZE (0..32))
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The group name of the user who the started process."
 ::= { psTableEntry 5 }

psProgramName OBJECT-TYPE
    SYNTAX DisplayString (SIZE (0..32))
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The name of the currently executing command or program,
        if any, given to the process."
 ::= { psTableEntry 6 }

psInvokingProcessId OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "If this is a subprocess, this is the identifier of the process
        that created this process."
 ::= { psTableEntry 7 }

psState OBJECT-TYPE
    SYNTAX INTEGER {
        stopped(1),
        ready(2),
        prePage(3),
        memoryWait(4),
        waitShort(5),
        waitLong(6),
        postPurge(7),
        postPurgeReady(8),
        workingSetCalc(9),
        workingSetCalcReady(10)
    }
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Current state of the process."
 ::= { psTableEntry 8 }

psPriority OBJECT-TYPE

```

```

SYNTAX INTEGER
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "The process priority."
::= { psTableEntry 9 }

psCpuTime OBJECT-TYPE
SYNTAX TimeTicks
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "Actual execution time for process. (1/100s second)"
::= { psTableEntry 10 }

psPageFaultTime OBJECT-TYPE
SYNTAX TimeTicks
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "Accumulated time the processor has spent in page faults
    for this process. (1/100s second)"
::= { psTableEntry 11 }

psPageFaults OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "Number of page faults for the process since it was created."
::= { psTableEntry 12 }

psWorkingSetSize OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "The number of pages in the current working set of the process."
::= { psTableEntry 13 }

psDiskReads OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "The number of times the process has read data from the
    disk into an input buffer since the process was created.
    Does not include page fault accesses."
::= { psTableEntry 14 }

psDiskWrites OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "The number of times the process has written data from an
    output buffer to disk since the process was created."

```

```

 ::= { psTableEntry 15 }

psNumInstances OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "The number of copies of this process name running in this
        module."
 ::= { psTableEntry 16 }

psNumProgramInstances OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "The number of copies of this program name running in this module.
        If this name is empty, the value will be 0."
 ::= { psTableEntry 17 }

-- The user trap table is a circular buffer of user trap messages from
-- all modules.  This table is not organized by module, rather the messages
-- are placed in the table in the order in which they arrive.  These
-- messages are accessed using a virtual index, userTrapMsgsNumber, a 32
-- bit unsigned number which keeps incrementing with every new entry.
-- The number of log items retained in this table is specified by the
-- configuration file variable UserTrapNum.  If this variable is not
-- set in the configuration file, this table will default to 40 log items.

userNumTrapMsgs OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "Virtual index of largest entry in user trap table."
 ::= { userinfo 1 }

userTrapMsgsTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF UserTrapMsgsTableEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
        "A table of messages from the user trap queue."
 ::= { userinfo 2 }

userTrapMsgsTableEntry OBJECT-TYPE
    SYNTAX  UserTrapMsgsTableEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
        "A row in the user trap message table."
    INDEX { userTrapMsgsNumber }
 ::= { userTrapMsgsTable 1 }

UserTrapMsgsTableEntry ::= SEQUENCE {
    userTrapMsgsNumber  INTEGER,
    userTrapMsgsMessage  OCTET STRING

```

}

userTrapMsgsNumber OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-only

STATUS mandatory

DESCRIPTION

"Index number assigned to this user trap message entry."

::= { userTrapMsgsTableEntry 1 }

userTrapMsgsMessage OBJECT-TYPE

SYNTAX OCTET STRING (SIZE (0..256))

ACCESS read-only

STATUS mandatory

DESCRIPTION

"User trap queue message."

::= { userTrapMsgsTableEntry 2 }

END