Industry-Standard Management Information Base

This chapter describes the industry-standard Management Information Base (MIB) text files that are supported by Unified Communications Manager and used with Simple Network Management Protocol (SNMP).

SYSAPPL-MIB

Note This is a reformatted version of SYSAPPL-MIB. Download and compile all of the MIBs in this section from http://tools.cisco.com/Support/SNMP/do/BrowseMIB.do?local=en&step=2.

The MIB module defines management objects that model applications as collections of executables and files installed and executing on a host system. The MIB presents a system-level view of applications; i.e., objects in this MIB are limited to those attributes that can typically be obtained from the system itself without adding special instrumentation to the applications.

Before you can compile SYSAPPL-MIB , you need to compile the MIBs listed below in the order listed.

1. RFC1155-SMI
2. RFC-1212
3. SNMPv2-SMI-v1
4. SNMPv2-TC-v1
5. SYSAPPL-MIB

Additional downloads are:

- OID File: SYSAPPL-MIB.oid

SYSAPPL-MIB Revisions

The following table lists the revisions to the MIS beginning with the latest revision.

Table 1: History of Revisions

Date	Action	Description
10-20-1997	IETF Applications MIB Working Group.	::= { mib-2 54 }

SYSAPPL-MIB Definitions

The following definitions are imported for SYSAPP-MIB:

- MODULE-IDENTITY, OBJECT-TYPE, mib-2, Unsigned32 (gotten from CISCO-TC for the time being until it becomes available in SNMPv2-SMI), Unsigned32, TimeTicks, Counter32, Gauge32 TimeTicks, Counter32, Gauge32
- From SNMPv2-SMI—Unsigned32
- From CISCO-TC—DateAndTime, TEXTUAL-CONVENTION
- From SNMPv2-TC—MODULE-COMPLIANCE, OBJECT-GROUP
- From SNMPv2-CONF;

System Application MIB

sysApplMIB MODULE-IDENTITY

sysApplOBJ OBJECT IDENTIFIER ::= { sysApplMIB 1 }

sysApplInstalled OBJECT IDENTIFIER ::= { sysApplOBJ 1 }

sysApplRun OBJECT IDENTIFIER ::= { sysApplOBJ 2 }

sysApplMap OBJECT IDENTIFIER ::= { sysApplOBJ 3 }

sysApplNotifications OBJECT IDENTIFIER ::= { sysApplMIB 2 }

sysApplConformance OBJECT IDENTIFIER ::= { sysApplMIB 3 }

System Application MIB Textual Conventions

RunState ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

This TC describes the current execution state of a running application or process. The possible values are: running(1), runnable(2), waiting for a resource (CPU, etc.) waiting(3), waiting for an event exiting(4), other(5) other invalid state.

SYNTAX INTEGER { running (1); runnable (2); waiting for resource and waiting (3); waiting for event and exiting (4); other (5) }

LongUtf8String ::= TEXTUAL-CONVENTION

DISPLAY-HINT 1024a

STATUS current

DESCRIPTION

To facilitate internationalization, this TC represents information taken from the ISO/IEC IS 10646-1 character set, encoded as an octet string using the UTF-8 character encoding scheme described in RFC 2044 [10]. For strings in 7-bit US-ASCII, there is no impact since the UTF-8 representation is identical to the US-ASCII encoding.

SYNTAX OCTET STRING (SIZE (0..1024))

Utf8String ::= TEXTUAL-CONVENTION

DISPLAY-HINT 255a

STATUS current

DESCRIPTION

To facilitate internationalization, this TC represents information taken from the ISO/IEC IS 10646-1 character set, encoded as an octet string using the UTF-8 character encoding scheme described in RFC 2044 [10]. For strings in 7-bit US-ASCII, there is no impact since the UTF-8 representation is identical to the US-ASCII encoding.

SYNTAX OCTET STRING (SIZE (0..255))

Installed Application Groups

This group provides information about application packages that have been installed on the host computer. The group contains two tables as follows:

- sysApplInstallPkgTable: Describes the application packages
- sysApplInstallElmtTable: Describes the constituent elements (files and executables) which compose an application package

In order to appear in the group, an application and its component files must be discoverable by the system itself, possibly through some type of software installation mechanism or registry.

sysApplInstallPkgTable

The system installed application packages table provides information on the software packages installed on a system. These packages may consist of many different files including executable and non-executable files.

sysApplInstallPkgTable OBJECT-TYPE

SYNTAX SysApplInstallPkgEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

The table listing the software application packages installed on a host computer. In order to appear in this table, it may be necessary for the application to be installed using some type of software installation mechanism or global registry so that its existence can be detected by the agent implementation.

::= { sysApplInstalled 1 }

sysApplInstallPkgEntry OBJECT-TYPE

SYNTAX SysApplInstallPkgEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

The logical row describing an installed application package.

INDEX{ sysApplInstallPkgIndex }

::= { sysApplInstallPkgTable 1 }

SysApplInstallPkgEntry ::= SEQUENCE { sysApplInstallPkgIndex Unsigned32, sysApplInstallPkgManufacturer Utf8String, sysApplInstallPkgProductName Utf8String, sysApplInstallPkgVersion Utf8String, sysApplInstallPkgSerialNumber Utf8String, sysApplInstallPkgDate DateAndTime, sysApplInstallPkgLocation LongUtf8String }

sysApplInstallPkgIndex OBJECT-TYPE

SYNTAX Unsigned32 (1..ffffffffh)

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

An integer used only for indexing purposes. Generally monotonically increasing from 1 as new applications are installed. The value for each installed application must remain constant at least from one re-initialization of the network management entity which implements this MIB module to the next re-initialization. The specific value is meaningful only within a given SNMP entity. A sysApplInstallPkgIndex value must not be re-used until the next agent entity restart in the event the installed application entry is deleted.

::= { sysApplInstallPkgEntry 1 }

sysApplInstallPkgManufacturer OBJECT-TYPE

SYNTAX Utf8String

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The Manufacturer of the software application package.

::= { sysApplInstallPkgEntry 2 }

sysApplInstallPkgProductName OBJECT-TYPE

SYNTAX Utf8String

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The name assigned to the software application package by the Manufacturer.

::= { sysApplInstallPkgEntry 3 }

sysApplInstallPkgVersion OBJECT-TYPE

SYNTAX Utf8String

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The version number assigned to the application package by the manufacturer of the software.

::= { sysApplInstallPkgEntry 4 }

sysApplInstallPkgSerialNumber OBJECT-TYPE

SYNTAX Utf8String

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The serial number of the software assigned by the manufacturer.

::= { sysApplInstallPkgEntry 5 }

sysApplInstallPkgDate OBJECT-TYPE

SYNTAX DateAndTime

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The date and time this software application was installed on the host.

::= { sysApplInstallPkgEntry 6 }

sysApplInstallPkgLocation OBJECT-TYPE

SYNTAX LongUtf8String

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The complete path name where the application package is installed. For example, the value would be `/opt/MyapplDir` if the application package was installed in the `/opt/MyapplDir` directory.

::= { sysApplInstallPkgEntry 7 }

sysApplInstallElmtTable

This table details the individual application package elements (files and executables) installed on the host computer which comprise the applications defined in the sysApplInstallPkg Table. Each entry in this table has an index to the sysApplInstallPkg table to identify the application package of which it is a part. As a result, there may be many entries in this table for each instance in the sysApplInstallPkg Table.

Table entries are indexed by sysApplInstallPkgIndex, sysApplInstallElmtIndex to facilitate retrieval of all elements associated with a particular installed application package.

sysApplInstallElmtTable OBJECT-TYPE

SYNTAX SEQUENCE OF SysApplInstallElmtEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

This table details the individual application package elements (files and executables) installed on the host computer which comprise the applications defined in the sysApplInstallPkg Table. Each entry in this table has an index to the sysApplInstallPkg table to identify the application package of which it is a part. As a result, there may be many entries in this table for each instance in the sysApplInstallPkg Table.

Table entries are indexed by sysApplInstallPkgIndex, sysApplInstallElmtIndex to facilitate retrieval of all elements associated with a particular installed application package.

::= { sysApplInstalled 2 }

sysApplInstallElmtEntry OBJECT-TYPE

SYNTAX SysApplInstallElmtEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

The logical row describing an element of an installed application. The element may be an executable or non-executable file.

INDEX {sysApplInstallPkgIndex, sysApplInstallElmtIndex}

::= { sysApplInstallElmtTable 1 }

SysApplInstallElmtEntry ::= SEQUENCE { sysApplInstallElmtIndex Unsigned32, sysApplInstallElmtNameUtf8String, sysApplInstallElmtTypeINTEGER, sysApplInstallElmtDateDateAndTime, sysApplInstallElmtPathLongUtf8String, sysApplInstallElmtSizeHighUnsigned32, sysApplInstallElmtSizeLow Unsigned32, sysApplInstallElmtRoleBITS, sysApplInstallElmtRoleOCTET STRING, sysApplInstallElmtModifyDate DateAndTime, sysApplInstallElmtCurSizeHighUnsigned32, sysApplInstallElmtCurSizeLow Unsigned32 }

sysApplInstallElmtIndex OBJECT-TYPE

SYNTAX Unsigned32 (1...ffffffffh)

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

An arbitrary integer used for indexing. The value of this index is unique among all rows in this table that exist or have existed since the last agent restart.

::= { sysApplInstallElmtEntry 1 }

sysApplInstallElmtName OBJECT-TYPE

SYNTAX Utf8String

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The name of this element which is contained in the application.

::= { sysApplInstallElmtEntry 2 }

sysApplInstallElmtType OBJECT-TYPE

SYNTAX INTEGER { unknown(1), nonexecutable(2), operatingSystem(3), executable deviceDriver(4), executable application(5), executable }

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The type of element that is part of the installed application.

::= { sysApplInstallElmtEntry 3 }

sysApplInstallElmtDate OBJECT-TYPE

SYNTAX DateAndTime

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The date and time that this component was installed on the system.

::= { sysApplInstallElmtEntry 4 }

sysApplInstallElmtPath OBJECT-TYPE

SYNTAX LongUtf8String

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The full directory path where this element is installed. For example, the value would be `/opt/EMPuma/bin` for an element installed in the directory `/opt/EMPuma/bin`. Most application packages include information about the elements contained in the package. In addition, elements are typically installed in sub-directories under the package installation directory. In cases where the element path names are not included in the package information itself, the path can usually be determined by a

simple search of the sub-directories. If the element is not installed in that location and there is no other information available to the agent implementation, then the path is unknown and null is returned.

::= { sysApplInstallElmtEntry 5}

sysApplInstallElmtSizeHigh OBJECT-TYPE

SYNTAX Unsigned32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The installed file size in 2^32 byte blocks. This is the size of the file on disk immediately after installation. For example, for a file with a total size of 4,294,967,296 bytes, this variable would have a value of 1; for a file with a total size of 4,294,967,295 bytes this variable would be 0.

::= { sysApplInstallElmtEntry 6 }

sysApplInstallElmtSizeLow OBJECT-TYPE

SYNTAX Unsigned32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The installed file size modulo 2^32 bytes. This is the size of the file on disk immediately after installation. For example, for a file with a total size of 4,294,967,296 bytes this variable would have a value of 0; for a file with a total size of 4,294,967,295 bytes this variable would be 4,294,967,295.

::= { sysApplInstallElmtEntry 7 }

sysApplInstallElmtRole OBJECT-TYPE

SYNTAX OCTET STRING (SIZE(1))

SYNTAX BITS { executable (0), exclusive (1), primary (2), required (3), dependent (4), unknown(5) }

MAX-ACCESS read-write

STATUS current

DESCRIPTION

An operator assigned value used in the determination of application status. This value is used by the agent to determine both the mapping of started processes to the initiation of an application, as well as to allow for a determination of application health. The default value, unknown(5), is used when an operator has not yet assigned one of the other values. If unknown(5) is set, bits 1 - 4 have no meaning. The possible values are:

- executable (0)—An application may have one or more executable elements. The rest of the bits have no meaning if the element is not executable.
- exclusive(1)—Only one copy of an exclusive element may be running per invocation of the running application.
- primary(2)—The primary executable. An application can have one, and only one element that is designated as the primary executable. The execution of this element constitutes an invocation of the application. This is used by the agent implementation to determine the initiation of an application.

The primary executable must remain running long enough for the agent implementation to detect its presence.

- required(3)—An application may have zero or more required elements. All required elements must be running in order for the application to be judged to be running and healthy.
- dependent(4)—An application may have zero or more dependent elements. Dependent elements may not be running unless required elements are.
- unknown(5)—Default value for the case when an operator has not yet assigned one of the other values. When set, bits 1, 2, 3, and 4 have no meaning.

sysApplInstallElmtRole is used by the agent implementation in determining the initiation of an application, the current state of a running application (see sysApplRunCurrentState), when an application invocation is no longer running, and the exit status of a terminated application invocation (see sysApplPastRunExitState).

--DEFVAL { 5 }

::= { sysApplInstallElmtEntry 8 }

sysApplInstallElmtModifyDate OBJECT-TYPE

SYNTAX DateAndTime

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The date and time that this element was last modified. Modification of the sysApplInstallElmtRole columnar object does NOT constitute a modification of the element itself and should not affect the value of this object.

::= { sysApplInstallElmtEntry 9 }

sysApplInstallElmtCurSizeHigh OBJECT-TYPE

SYNTAX Unsigned32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The current file size in 2^32 byte blocks. For example, for a file with a total size of 4,294,967,296 bytes, this variable would have a value of 1; for a file with a total size of 4,294,967,295 bytes this variable would be 0.

::= { sysApplInstallElmtEntry 10 }

sysApplInstallElmtCurSizeLow OBJECT-TYPE

SYNTAX Unsigned32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The current file size modulo 2^32 bytes. For example, for a file with a total size of 4,294,967,296 bytes this variable would have a value of 0; for a file with a total size of 4,294,967,295 bytes this variable would be 4,294,967,295.

::= { sysApplInstallElmtEntry 11 }

sysApplRun Group

This group models activity information for applications that have been invoked and are either currently running, or have previously run on the host system. Likewise, the individual elements of an invoked application are also modeled to show currently running processes, and processes that have run in the past.

sysApplRunTable

The sysApplRunTable contains the application instances which are currently running on the host. Since a single application might be invoked multiple times, an entry is added to this table for each INVOCATION of an application. The table is indexed by sysApplInstallPkgIndex, sysApplRunIndex to enable managers to easily locate all invocations of a particular application package.

sysApplRunTable OBJECT-TYPE

SYNTAX SEQUENCE OF SysApplRunEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

The table describes the applications which are executing on the host. Each time an application is invoked, an entry is created in this table. When an application ends, the entry is removed from this table and a corresponding entry is created in the SysApplPastRunTable.

A new entry is created in this table whenever the agent implementation detects a new running process that is an installed application element whose sysApplInstallElmtRole designates it as being the application's primary executable (sysApplInstallElmtRole = primary(2)).

The table is indexed by sysApplInstallPkgIndex, sysApplRunIndex to enable managers to easily locate all invocations of a particular application package.

::= { sysApplRun 1 }

sysApplRunEntry OBJECT-TYPE

SYNTAX SysApplRunEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

The logical row describing an application which is currently running on this host.

INDEX { sysApplInstallPkgIndex, sysApplRunIndex }

::= { sysApplRunTable 1 }

SysApplRunEntry ::= SEQUENCE { sysApplRunIndex Unsigned32, sysApplRunStarted DateAndTime, sysApplRunCurrentState RunState }

sysApplRunIndex OBJECT-TYPE

SYNTAX Unsigned32 (1..'ffffffff'h)

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

Part of the index for this table. An arbitrary integer used only for indexing purposes. Generally monotonically increasing from 1 as new applications are started on the host, it uniquely identifies application invocations.

The numbering for this index increases by 1 for each INVOCATION of an application, regardless of which installed application package this entry represents a running instance of. An example of the indexing for a couple of entries is shown below.

sysApplRunStarted.17.14

sysApplRunStarted.17.63

sysApplRunStarted.18.13

:

In this example, the agent has observed 12 application invocations when the application represented by entry 18 in the sysApplInstallPkgTable is invoked. The next invocation detected by the agent is an invocation of installed application package 17. Some time later, installed application 17 is invoked a second time.

Note This index is not intended to reflect a real-time (wall clock time) ordering of application invocations; it is merely intended to uniquely identify running instances of applications. Although the sysApplInstallPkgIndex is included in the INDEX clause for this table, it serves only to ease searching of this table by installed application and does not contribute to uniquely identifying table entries.

::= { sysApplRunEntry 1 }

sysApplRunStarted OBJECT-TYPE

SYNTAX DateAndTime

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The date and time that the application was started.

::= { sysApplRunEntry 2 }

sysApplRunCurrentState OBJECT-TYPE

SYNTAX RunState

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The current state of the running application instance. The possible values are running(1), runnable(2) but waiting for a resource such as CPU, waiting(3) for an event, exiting(4), or other(5). This value is based on an evaluation of the running elements of this application instance (see sysApplElmRunState) and their Roles as defined by sysApplInstallElmtRole. An agent implementation may detect that an application instance is in the process of exiting if one or more of its REQUIRED elements are no longer running. Most agent implementations will wait until a second internal poll has been completed to give the system time to start REQUIRED elements before marking the application instance as exiting.

::= { sysApplRunEntry 3 }

sysApplPastRunTable

The sysApplPastRunTable provides a history of applications previously run on the host computer. Entries are removed from the sysApplRunTable and corresponding entries are added to this table when an application becomes inactive. Entries remain in this table until they are aged out when either the table size reaches a maximum as determined by the sysApplPastRunMaxRows, or when an entry has aged to exceed a time limit as set be sysApplPastRunTblTimeLimit.

When aging out entries, the oldest entry, as determined by the value of sysApplPastRunTimeEnded, will be removed first.

sysApplPastRunTable OBJECT-TYPE

SYNTAX SEQUENCE OF SysApplPastRunEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

A history of the applications that have previously run on the host computer. An entry's information is moved to this table from the sysApplRunTable when the invoked application represented by the entry ceases to be running. An agent implementation can determine that an application invocation is no longer running by evaluating the running elements of the application instance and their Roles as defined by sysApplInstallElmtRole. Obviously, if there are no running elements for the application instance, then the application invocation is no longer running.

If any one of the REQUIRED elements is not running, the application instance may be in the process of exiting. Most agent implementations will wait until a second internal poll has been completed to give the system time to either restart partial failures or to give all elements time to exit. If, after the second poll, there are REQUIRED elements that are not running, then the application instance may be considered by the agent implementation to no longer be running.

Entries remain in the sysApplPastRunTable until they are aged out when either the table size reaches a maximum as determined by the sysApplPastRunMaxRows, or when an entry has aged to exceed a time limit as set by sysApplPastRunTblTimeLimit.

Entries in this table are indexed by sysApplInstallPkgIndex, sysApplPastRunIndex to facilitate retrieval of all past run invocations of a particular installed application.

::= { sysApplRun 2 }

sysApplPastRunEntry OBJECT-TYPE

SYNTAX SysApplPastRunEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

The logical row describing an invocation of an application which was previously run and has terminated. The entry is basically copied from the sysApplRunTable when the application instance terminates. Hence, the entry's value for sysApplPastRunIndex is the same as its value was for sysApplRunIndex.

INDEX{ sysApplInstallPkgIndex, sysApplPastRunIndex }

::= { sysApplPastRunTable 1 }

SysApplPastRunEntry ::= SEQUENCE { sysApplPastRunIndex Unsigned32, sysApplPastRunStarted DateAndTime, sysApplPastRunExitState INTEGER, sysApplPastRunTimeEnded DateAndTime

sysApplPastRunIndex OBJECT-TYPE

SYNTAX Unsigned32 (1...ffffffffh)

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

Part of the index for this table. An integer matching the value of the removed sysApplRunIndex corresponding to this row.

::= { sysApplPastRunEntry 1 }

sysApplPastRunStarted OBJECT-TYPE

SYNTAX DateAndTime

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The date and time that the application was started.

::= { sysApplPastRunEntry 2 }

sysApplPastRunExitState OBJECT-TYPE

SYNTAX INTEGER { complete (1), failed (2), other (3) }

- complete (1)—normal exit at sysApplRunTimeEnded
- failed (2)—abnormal exit
- other (3)

  MAX-ACCESS read-only

  STATUS current

  DESCRIPTION

  The state of the application instance when it terminated. This value is based on an evaluation of the running elements of an application and their Roles as defined by sysApplInstallElmtRole. An application instance is said to have exited in a COMPLETE state and its entry is removed from the sysApplRunTable and added to the sysApplPastRunTable when the agent detects that ALL elements of an application invocation are no longer running. Most agent implementations will wait until a

second internal poll has been completed to give the system time to either restart partial failures or to give all elements time to exit. A failed state occurs if, after the second poll, any elements continue to run but one or more of the REQUIRED elements are no longer running.

All other combinations MUST be defined as OTHER.

::= { sysApplPastRunEntry 3 }

sysApplPastRunTimeEnded OBJECT-TYPE

SYNTAX DateAndTime

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The DateAndTime the application instance was determined to be no longer running.

::= { sysApplPastRunEntry 4 }

sysApplElmtRunTable

The sysApplElmtRunTable contains an entry for each process that is currently running on the host. An entry is created in this table for each process at the time it is started, and will remain in the table until the process terminates. The table is indexed by sysApplElmtRunInstallPkg, sysApplElmtRunInvocID, and sysApplElmtRunIndex to make it easy to locate all running elements of a particular invoked application which has been installed on the system.

sysApplElmtRunTable OBJECT-TYPE

SYNTAX SEQUENCE OF SysApplElmtRunEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

The table describes the processes which are currently executing on the host system. Each entry represents a running process and is associated with the invoked application of which that process is a part, if possible. This table contains an entry for every process currently running on the system, regardless of whether its 'parent' application can be determined. So, for example, processes like 'ps' and 'grep' will have entries though they are not associated with an installed application package.

Because a running application may involve more than one executable, it is possible to have multiple entries in this table for each application. Entries are removed from this table when the process terminates. The table is indexed by sysApplElmtRunInstallPkg, sysApplElmtRunInvocID, and sysApplElmtRunIndex to facilitate the retrieval of all running elements of a particular invoked application which has been installed on the system.

::= { sysApplRun 3 }

sysApplElmtRunEntry OBJECT-TYPE

SYNTAX SysApplElmtRunEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

The logical row describing a process currently running on this host. When possible, the entry is associated with the invoked application of which it is a part.

INDEX{ sysApplElmtRunInstallPkg, sysApplElmtRunInvocID, sysApplElmtRunIndex }

::= { sysApplElmtRunTable 1 }

SysApplElmtRunEntry ::= SEQUENCE {sysApplElmtRunInstallPkg Unsigned32, sysApplElmtRunInvocIDUnsigned32, sysApplElmtRunIndex Unsigned32, sysApplElmtRunInstallID Unsigned32, sysApplElmtRunTimeStartedDateAndTime, sysApplElmtRunState RunState, sysApplElmtRunNameLongUtf8String, sysApplElmtRunParameters Utf8String, sysApplElmtRunCPU TimeTicks, sysApplElmtRunMemory Gauge32, sysApplElmtRunNumFiles Gauge32, sysApplElmtRunUserUtf8String }

sysApplElmtRunInstallPkg OBJECT-TYPE

SYNTAX Unsigned32 (0...ffffffffh)

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

Part of the index for this table, this value identifies the installed software package for the application of which this process is a part. Provided that the process's 'parent' application can be determined, the value of this object is the same value as the sysApplInstallPkgIndex for the entry in the sysApplInstallPkgTable that corresponds to the installed application of which this process is a part.

If, however, the 'parent' application cannot be determined, (for example the process is not part of a particular installed application), the value for this object is then '0', signifying that this process cannot be related back to an application, and in turn, an installed software package.

::= { sysApplElmtRunEntry 1 }

sysApplElmtRunInvocID OBJECT-TYPE

SYNTAX Unsigned32 (0...ffffffffh)

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

Part of the index for this table, this value identifies the invocation of an application of which this process is a part. Provided that the 'parent' application can be determined, the value of this object is the same value as the sysApplRunIndex for the corresponding application invocation in the sysApplRunTable.

If, however, the 'parent' application cannot be determined, the value for this object is then '0', signifying that this process cannot be related back to an invocation of an application in the sysApplRunTable.

::= { sysApplElmtRunEntry 2 }

sysApplElmtRunIndex OBJECT-TYPE

SYNTAX Unsigned32 (0...ffffffffh)

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

Part of the index for this table. A unique value for each process running on the host. Wherever possible, this should be the system's native, unique identification number.

::= { sysApplElmtRunEntry 3 }

sysApplElmtRunInstallID OBJECT-TYPE

SYNTAX Unsigned32 (0...ffffffffh)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The index into the sysApplInstallElmtTable. The value of this object is the same value as the sysApplInstallElmtIndex for the application element of which this entry represents a running instance.

If this process cannot be associated with an installed executable, the value should be '0'.

::= { sysApplElmtRunEntry 4 }

sysApplElmtRunTimeStarted OBJECT-TYPE

SYNTAX DateAndTime

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The time the process was started.

::= { sysApplElmtRunEntry 5 }

sysApplElmtRunState OBJECT-TYPE

SYNTAX RunState

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The current state of the running process. The possible values are running(1), runnable(2) but waiting for a resource such as CPU, waiting(3) for an event, exiting(4), or other(5).

::= { sysApplElmtRunEntry 6 }

sysApplElmtRunName OBJECT-TYPE

SYNTAX LongUtf8String

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The full path and filename of the process. For example, `/opt/MYYpkg/bin/myyproc` would be returned for process myyproc whose execution path is `/opt/MYYpkg/bin/myyproc`.

::= { sysApplElmtRunEntry 7 }

sysApplElmtRunParameters OBJECT-TYPE

SYNTAX Utf8String

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The starting parameters for the process.

::= { sysApplElmtRunEntry 8 }

sysApplElmtRunCPU OBJECT-TYPE

SYNTAX TimeTicks

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The number of centi-seconds of the total system CPU resources consumed by this process. Note that on a multi-processor system, this value may have been incremented by more than one centi-second in one centi-second of real (wall clock) time.

::= { sysApplElmtRunEntry 9 }

sysApplElmtRunMemory OBJECT-TYPE

SYNTAX Gauge32

UNITS Kbytes

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The total amount of real system memory measured in Kbytes currently allocated to this process.

::= { sysApplElmtRunEntry 10 }

sysApplElmtRunNumFiles OBJECT-TYPE

SYNTAX Gauge32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The number of regular files currently open by the process. Transport connections (sockets) should NOT be included in the calculation of this value, nor should operating system specific special file types.

::= { sysApplElmtRunEntry 11 }

sysApplElmtRunUser OBJECT-TYPE

SYNTAX Utf8String

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The process owner's login name (e.g. root).

::= { sysApplElmtRunEntry 12 }

sysApplElmtPastRunTable

The sysApplElmtPastRunTable maintains a history of processes which have previously executed on the host as part of an application. Upon termination of a process, the entry representing the process is removed from the sysApplElmtRunTable and a corresponding entry is created in this table provided that the process was part of an identifiable application. If the process could not be associated with an invoked application, no corresponding entry is created.

Hence, whereas the sysApplElmtRunTable contains an entry for every process currently executing on the system, the sysApplElmtPastRunTable only contains entries for processes that previously executed as part of an invoked application.

Entries remain in this table until they are aged out when either the number of entries in the table reaches a

maximum as determined by sysApplElmtPastRunMaxRows, or when an entry has aged to exceed a time limit as set by sysApplElmtPastRunTblTimeLimit. When aging out entries, the oldest entry, as determined by the value of sysApplElmtPastRunTimeEnded, will be removed first.

The table is indexed by sysApplInstallPkgIndex (from the sysApplInstallPkgTable), sysApplElmtPastRunInvocID, and sysApplElmtPastRunIndex to make it easy to locate all previously executed processes of a particular invoked application that has been installed on the system.

sysApplElmtPastRunTable OBJECT-TYPE

SYNTAX SEQUENCE OF SysApplElmtPastRunEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

The table describes the processes which have previously executed on the host system as part of an application. Each entry represents a process which has previously executed and is associated with the invoked application of which it was a part. Because an invoked application may involve more than one executable, it is possible to have multiple entries in this table for each application invocation. Entries are added to this table when the corresponding process in the sysApplElmtRun Table terminates.

Entries remain in this table until they are aged out when either the number of entries in the table reaches a maximum as determined by sysApplElmtPastRunMaxRows, or when an entry has aged to exceed a time limit as set by sysApplElmtPastRunTblTimeLimit. When aging out entries, the oldest entry, as determined by the value of sysApplElmtPastRunTimeEnded, will be removed first.

The table is indexed by sysApplInstallPkgIndex (from the sysApplInstallPkgTable), sysApplElmtPastRunInvocID, and sysApplElmtPastRunIndex to make it easy to locate all previously executed processes of a particular invoked application that has been installed on the system.

::= { sysApplRun 4 }

sysApplElmtPastRunEntry OBJECT-TYPE

SYNTAX SysApplElmtPastRunEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

The logical row describing a process which was previously executed on this host as part of an installed application. The entry is basically copied from the sysApplElmtRunTable when the process terminates. Hence, the entry's value for sysApplElmtPastRunIndex is the same as its value was for sysApplElmtRunIndex. Note carefully: only those processes which could be associated with an identified application are included in this table.

INDEX{ sysApplInstallPkgIndex, sysApplElmtPastRunInvocID, sysApplElmtPastRunIndex }

::= { sysApplElmtPastRunTable 1 }

SysApplElmtPastRunEntry ::= SEQUENCE { sysApplElmtPastRunInvocIDUnsigned32, sysApplElmtPastRunIndex Unsigned32, sysApplElmtPastRunInstallID Unsigned32, sysApplElmtPastRunTimeStartedDateAndTime, sysApplElmtPastRunTimeEnded DateAndTime, sysApplElmtPastRunNameLongUtf8String, sysApplElmtPastRunParameters Utf8String, sysApplElmtPastRunCPU TimeTicks, sysApplElmtPastRunMemory Unsigned32, sysApplElmtPastRunNumFiles Unsigned32, sysApplElmtPastRunUserUtf8String }

sysApplElmtPastRunInvocID OBJECT-TYPE

SYNTAX Unsigned32 (1...ffffffffh)

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

Part of the index for this table, this value identifies the invocation of an application of which the process represented by this entry was a part. The value of this object is the same value as the sysApplRunIndex for the corresponding application invocation in the sysApplRunTable. If the invoked application as a whole has terminated, it will be the same as the sysApplPastRunIndex.

::= { sysApplElmtPastRunEntry 1 }

sysApplElmtPastRunIndex OBJECT-TYPE

SYNTAX Unsigned32 (0..'ffffffff'h)

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

Part of the index for this table. An integer assigned by the agent equal to the corresponding sysApplElmtRunIndex which was removed from the sysApplElmtRunTable and moved to this table when the element terminated. Note that entries in this table are indexed by sysApplElmtPastRunInvocID, sysApplElmtPastRunIndex.

The possibility exists, though unlikely, of a collision occurring by a new entry which was run by the same invoked application (InvocID), and was assigned the same process identification number (ElmtRunIndex) as an element which was previously run by the same invoked application.

Should this situation occur, the new entry replaces the old entry.

See the Implementation Issues section, sysApplElmtPastRunTable Entry Collisions for the conditions that would have to occur in order for a collision to occur.

::= { sysApplElmtPastRunEntry 2 }

sysApplElmtPastRunInstallID OBJECT-TYPE

SYNTAX Unsigned32 (1..'ffffffff'h)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The index into the installed element table. The value of this object is the same value as the sysApplInstallElmtIndex for the application element of which this entry represents a previously executed process.

::= { sysApplElmtPastRunEntry 3 }

sysApplElmtPastRunTimeStarted OBJECT-TYPE

SYNTAX DateAndTime

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The time the process was started.

::= { sysApplElmtPastRunEntry 4 }

sysApplElmtPastRunTimeEnded OBJECT-TYPE

SYNTAX DateAndTime

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The time the process ended.

::= { sysApplElmtPastRunEntry 5 }

sysApplElmtPastRunName OBJECT-TYPE

SYNTAX LongUtf8String

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The full path and filename of the process. For example, '`/opt/MYYpkg/bin/myyproc`' would be returned for process 'myyproc' whose execution path was '`/opt/MYYpkg/bin/myyproc`'.

::= { sysApplElmtPastRunEntry 6 }

sysApplElmtPastRunParameters OBJECT-TYPE

SYNTAX Utf8String

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The starting parameters for the process.

::= { sysApplElmtPastRunEntry 7 }

sysApplElmtPastRunCPU OBJECT-TYPE

SYNTAX TimeTicks

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The last known number of centi-seconds of the total system's CPU resources consumed by this process. Note that on a multi-processor system, this value may increment by more than one centi-second in one centi-second of real (wall clock) time.

::= { sysApplElmtPastRunEntry 8 }

sysApplElmtPastRunMemory OBJECT-TYPE

SYNTAX Unsigned32 (0..'ffffffff'h)

UNITSKbytes

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The last known total amount of real system memory measured in Kbytes allocated to this process before it terminated.

::= { sysApplElmtPastRunEntry 9 }

sysApplElmtPastRunNumFiles OBJECT-TYPE

SYNTAX Unsigned32 (0..'ffffffff'h)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The last known number of files open by the process before it terminated. Transport connections (sockets) should NOT be included in the calculation of this value.

::= { sysApplElmtPastRunEntry 10 }

sysApplElmtPastRunUser OBJECT-TYPE

SYNTAX Utf8String

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The process owner's login name (e.g. root).

::= { sysApplElmtPastRunEntry 11 }

Additional Scalar Objects Controlling Table Sizes

sysApplPastRunMaxRows OBJECT-TYPE

SYNTAX Unsigned32 (0..'ffffffff'h)

MAX-ACCESS read-write

STATUS current

DESCRIPTION

The maximum number of entries allowed in the sysApplPastRunTable. Once the number of rows in the sysApplPastRunTable reaches this value, the management subsystem will remove the oldest entry in the table to make room for the new entry to be added. Entries will be removed on the basis of oldest sysApplPastRunTimeEnded value first.

This object may be used to control the amount of system resources that can used for sysApplPastRunTable entries. A conforming implementation should attempt to support the default value, however, a lesser value may be necessary due to implementation-dependent issues and resource availability.

DEFVAL { 500 }

::= { sysApplRun 5 }

sysApplPastRunTableRemItems OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

A counter of the number of entries removed from the sysApplPastRunTable because of table size limitations as set in sysApplPastRunMaxRows. This counter is the number of entries the management subsystem has had to remove in order to make room for new entries (so as not to exceed the limit set by sysApplPastRunMaxRows) since the last initialization of the management subsystem.

::= { sysApplRun 6 }

sysApplPastRunTblTimeLimit OBJECT-TYPE

SYNTAX Unsigned32 (0..'ffffffff'h)

UNITSseconds

MAX-ACCESS read-write

STATUS current

DESCRIPTION

The maximum time in seconds which an entry in the sysApplPastRunTable may exist before it is removed. Any entry that is older than this value will be removed (aged out) from the table. Note that an entry may be aged out prior to reaching this time limit if it is the oldest entry in the table and must be removed to make space for a new entry so as to not exceed sysApplPastRunMaxRows.

DEFVAL { 7200 }

::= { sysApplRun 7 }

sysApplElemPastRunMaxRows OBJECT-TYPE

SYNTAX Unsigned32 (0..'ffffffff'h)

MAX-ACCESS read-write

STATUS current

DESCRIPTION

The maximum number of entries allowed in the sysApplElmtPastRunTable. Once the number of rows in the sysApplElmtPastRunTable reaches this value, the management subsystem will remove the oldest entry to make room for the new entry to be added. Entries will be removed on the basis of oldest sysApplElmtPastRunTimeEnded value first. This object may be used to control the amount of system resources that can used for sysApplElemPastRunTable entries. A conforming implementation should attempt to support the default value, however, a lesser value may be necessary due to implementation-dependent issues and resource availability.

DEFVAL { 500 }

::= { sysApplRun 8 }

sysApplElemPastRunTableRemItems OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

A counter of the number of entries removed from the sysApplElemPastRunTable because of table size limitations as set in sysApplElemPastRunMaxRows. This counter is the number of entries the management subsystem has had to remove in order to make room for new entries (so as not to exceed the limit set by sysApplElemPastRunMaxRows) since the last initialization of the management subsystem.

::= { sysApplRun 9 }

sysApplElemPastRunTblTimeLimit OBJECT-TYPE

SYNTAX Unsigned32 (0..'ffffffff'h)

UNITSseconds

MAX-ACCESS read-write

STATUS current

DESCRIPTION

The maximum time in seconds which an entry in the sysApplElemPastRunTable may exist before it is removed. Any entry that is older than this value will be removed (aged out) from the table. Note that an entry may be aged out prior to reaching this time limit if it is the oldest entry in the table and must be removed to make space for a new entry so as to not exceed sysApplElemPastRunMaxRows.

DEFVAL { 7200 }

::= { sysApplRun 10 }

sysApplAgentPollInterval OBJECT-TYPE

SYNTAX Unsigned32 (0..'ffffffff'h)

UNITS seconds

MAX-ACCESS read-write

STATUS current

DESCRIPTION

The minimum interval in seconds that the management subsystem implementing this MIB will poll the status of the managed resources. Because of the non-trivial effort involved in polling the managed resources, and because the method for obtaining the status of the managed resources is implementation-dependent, a conformant implementation may chose a lower bound greater than 0.

A value of 0 indicates that there is no delay in the passing of information from the managed resources to the agent.

DEFVAL { 60 }

::= { sysApplRun 11 }

sysApplMap Group

This group contains a table, the sysApplMapTable, whose sole purpose is to provide a 'backwards' mapping so that, given a known sysApplElmtRunIndex (process identification number), the corresponding invoked application (sysApplRunIndex), installed element (sysApplInstallElmtIndex), and installed application package (sysApplInstallPkgIndex) can be quickly determined. The table will contain one entry for each process currently running on the system.

A backwards mapping is extremely useful since the tables in this MIB module are typically indexed with the installed application package (sysApplInstallPkgIndex) as the primary key, and on down as required by the specific table, with the process ID number (sysApplElmtRunIndex) being the least significant key.

It is expected that management applications will use this mapping table by doing a 'GetNext' operation with the known process ID number (sysApplElmtRunIndex) as the partial instance identifier. Assuming that there is an entry for the process, the result should return a single columnar value, the sysApplMapInstallPkgIndex, with the sysApplElmtRunIndex, sysApplRunIndex, and sysApplInstallElmtIndex contained in the instance identifier for the returned MIB object value.

Note If the process can not be associated back to an invoked application installed on the system, then the value returned for the columnar value sysApplMapInstallPkgIndex will be '0' and the instance portion of the object-identifier will be the process ID number (sysApplElmtRunIndex) followed by 0.0.

sysApplMapTable OBJECT-TYPE

SYNTAX SEQUENCE OF SysApplMapEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

The sole purpose of this table is to provide a 'backwards' mapping so that, given a known sysApplElmtRunIndex (process identification number), the corresponding invoked application (sysApplRunIndex), installed element (sysApplInstallElmtIndex), and installed application package (sysApplInstallPkgIndex) can be quickly determined.

::= { sysApplMap 1 }

sysApplMapEntry OBJECT-TYPE

SYNTAX SysApplMapEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

A logical row representing a process currently running on the system. This entry provides the index mapping from process identifier, back to the invoked application, installed element, and finally, the installed application package. The entry includes only one accessible columnar object, the sysApplMapInstallPkgIndex, but the invoked application and installed element can be determined from the instance identifier since they form part of the index clause.

INDEX { sysApplElmtRunIndex, sysApplElmtRunInvocID, sysApplMapInstallElmtIndex }

SysApplMapEntry ::= SEQUENCE { sysApplMapInstallElmtIndexUnsigned32, sysApplMapInstallPkgIndex Unsigned32 }

::= { sysApplMapTable 1 }

sysApplMapInstallElmtIndex OBJECT-TYPE

SYNTAX Unsigned32 (0..'ffffffff'h)

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

The index into the sysApplInstallElmtTable. The value of this object is the same value as the sysApplInstallElmtIndex for the application element of which this entry represents a running instance. If this process cannot be associated to an installed executable, the value should be '0'.

::= { sysApplMapEntry 1 }

sysApplMapInstallPkgIndex OBJECT-TYPE

SYNTAX Unsigned32 (0..'ffffffff'h)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The value of this object identifies the installed software package for the application of which this process is a part. Provided that the process's 'parent' application can be determined, the value of this object is the same value as the sysApplInstallPkgIndex for the entry in the sysApplInstallPkgTable that corresponds to the installed application of which this process is a part.

If, however, the 'parent' application cannot be determined, (for example the process is not part of a particular installed application), the value for this object is then '0', signifying that this process cannot be related back to an application, and in turn, an installed software package.

::= { sysApplMapEntry 2 }

Conformance Macros

sysApplMIBCompliances OBJECT IDENTIFIER

::= { sysApplConformance 1 }

sysApplMIBGroupsOBJECT IDENTIFIER

::= { sysApplConformance 2 }

sysApplMIBCompliance MODULE-COMPLIANCE

STATUS current

DESCRIPTION

Describes the requirements for conformance to the System Application MIB MODULE.

MANDATORY-GROUPS { sysApplInstalledGroup, sysApplRunGroup, sysApplMapGroup }

::= { sysApplMIBCompliances 1 }

sysApplInstalledGroup OBJECT-GROUP

OBJECTS { sysApplInstallPkgManufacturer, sysApplInstallPkgProductName, sysApplInstallPkgVersion, sysApplInstallPkgSerialNumber, sysApplInstallPkgDate, sysApplInstallPkgLocation, sysApplInstallElmtName, sysApplInstallElmtType, sysApplInstallElmtDate, sysApplInstallElmtPath, sysApplInstallElmtSizeHigh, sysApplInstallElmtSizeLow, sysApplInstallElmtRole, sysApplInstallElmtModifyDate, sysApplInstallElmtCurSizeHigh, sysApplInstallElmtCurSizeLow }

STATUS current

DESCRIPTION

The system application installed group contains information about applications and their constituent components which have been installed on the host system.

::= { sysApplMIBGroups 1 }

sysApplRunGroup OBJECT-GROUP

OBJECTS { sysApplRunStarted, sysApplRunCurrentState, sysApplPastRunStarted, sysApplPastRunExitState, sysApplPastRunTimeEnded, sysApplElmtRunInstallID, sysApplElmtRunTimeStarted, sysApplElmtRunState, sysApplElmtRunName, sysApplElmtRunParameters, sysApplElmtRunCPU, sysApplElmtRunMemory, sysApplElmtRunNumFiles, sysApplElmtRunUser, sysApplElmtPastRunInstallID, sysApplElmtPastRunTimeStarted, sysApplElmtPastRunTimeEnded, sysApplElmtPastRunName, sysApplElmtPastRunParameters, sysApplElmtPastRunCPU, sysApplElmtPastRunMemory, sysApplElmtPastRunNumFiles, sysApplElmtPastRunUser, sysApplPastRunMaxRows, sysApplPastRunTableRemItems, sysApplPastRunTblTimeLimit, sysApplElemPastRunMaxRows, sysApplElemPastRunTableRemItems, sysApplElemPastRunTblTimeLimit, sysApplAgentPollInterval }

STATUS current

DESCRIPTION

The system application run group contains information about applications and associated elements which have run or are currently running on the host system.

::= { sysApplMIBGroups 2 }

sysApplMapGroup OBJECT-GROUP

OBJECTS { sysApplMapInstallPkgIndex }

STATUS current

DESCRIPTION

The Map Group contains a single table, sysApplMapTable, that provides a backwards mapping for determining the invoked application, installed element, and installed application package given a known process identification number.

::= { sysApplMIBGroups 3 }

Troubleshoot System Application MIB

Linux and Cisco Unified CM Releases 5.x 6.x 7.x

Collect the following logs and information for analysis. Execute the command **file get activelog** *<paths below>*

- SNMP Master Agent Path : `/platform/snmp/snmpdm/*`
- System Application Agent Path: `/platform/snmp/sappagt/*`

Windows and Cisco Unified CM Release 4.x

Collect the following logs and information for analysis:

- Set the sysapp trace level to Detailed as follows, Enable TraceEnabled to “true” and TraceLevel to 3 from Registry HKEY_LOCAL_MACHINE\SOFTWARE\Cisco Systems, Inc.\SnmpSysAppAgent.
- Once you have edited it, restart the SNMP Service from the Services tab. You will see a trace file `C:\Program Files\Cisco\bin\SnmpSysAppImpl.log` created.
- Run a snmpwalk on the sysApplInstallPkgTable.
- Run a snmpwalk on the SysApplRunTable.
- Collect the `C:\Program Files\Cisco\bin\SnmpSysAppImpl.log` log file once walk is completed.
- Collect the application and event logs from the event log viewer.

Servlets for Cisco Unified CM 7.x

The SysAppl MIB provides a way to get inventory of what is installed and running at a given time. SysAppl agent cannot give the list of services activated or deactivated. It can only provide the running/not running states of the application/services. Web App services/Servlets cannot be monitored using the SysAppl MIB. Following are servlets for a 7.x system:

- Cisco CallManager Admin

- Cisco CallManager Cisco IP Phone Services
- Cisco CallManager Personal Directory
- Cisco CallManager Serviceability
- Cisco CallManager Serviceability RTMT
- Cisco Dialed Number Analyzer
- Cisco Extension Mobility
- Cisco Extension Mobility Application
- Cisco RTMT Reporter Servlet
- Cisco Tomcat Stats Servlet
- Cisco Trace Collection Servlet
- Cisco AXL Web Service
- Cisco Unified Mobile Voice Access Service
- Cisco Extension Mobility
- Cisco IP Manager Assistant
- Cisco WebDialer Web Service
- Cisco CAR Web Service
- Cisco Dialed Number Analyzer

For monitoring important service status for system health purposes, the following approaches are recommended:

- Use the Serviceability API called GetServiceStatus. This API can provide complete status information including activation status for both web application type and non web app services. (See AXL Serviceability API Guide for more details.)
- Use the **utils service list** command to check the status of different services.
- Use the Syslog message and monitor the servM generated messages. For example:

```
Mar 18 16:40:52 ciscart26 local7 6 : 92: Mar 18 11:10:52.630 UTC :
%CCM_SERVICEMANAGER-SERVICEMANAGER-6-ServiceActivated: Service Activated.
Service Name:Cisco CallManager SNMP Service App ID:Cisco Service Manager
Cluster ID: Node ID:ciscart26
```

Frequently Asked Questions for System Application MIB

When the CCMVersion MIB and sysApplRunCurrentState returns incorrect values in Cisco Unified CM Release 4.x, refer to CSCsk74156 to check if it is being hit. Verify if the fix for the defect has gone into the Cisco Unified CM version used by customer.

When the SNMP walk on sysApp MIB is not responding, refer to CSCsh72473 to check if it is being hit. Verify if the fix for the defect has gone into the Cisco Unified CM version used by customer.

RFC1213-MIB (MIB-II)

Note This is a reformatted version of MIB-II. Download and compile all of the MIBs in this section from http://tools.cisco.com/Support/SNMP/do/BrowseMIB.do?local=en&step=2

Before you can compile RFC1213-MIB, you need to compile the MIBs listed below in the order listed.

1. SNMPv2-SMI
2. SNMPv2-TC
3. IANAifType-MIB
4. RFC1155-SMI
5. RFC-1212
6. RFC1213-MIB

RFC1213-MIB Revisions

The following changes have been applied:

- The enumerations unknown(4) and dormant(5) have been added to ifOperStatus to reflect a change to the ifTable introduced in RFC 1573.
- The SYNTAX of ifType has been changed to IANAifType, to reflect the change to the ifTable introduced in RFC1573.

RFC1213-MIB Definitions

The following definitions are imported for MIB-II:

- mgmt, NetworkAddress, IpAddress, Counter, Gauge, TimeTicks
- From RFC1155-SMI—OBJECT-TYPE
- From RFC-1212—TEXTUAL-CONVENTION
- From SNMPv2-TC—IANAifType
- From IANAifType-MIB;

RFC1213-MIB Object Identifiers

This MIB module uses the extended OBJECT-TYPE macro as defined in [14]. MIB-II (same prefix as MIB-I) mib-2 OBJECT IDENTIFIER ::= { mgmt 1 }.

RFC1213-MIB Textual Conventions

DisplayString ::= OCTET STRING

This data type is used to model textual information taken from the NVT ASCII character set. By convention, objects with this syntax are declared as having SIZE (0..255).

PhysAddress ::= OCTET STRING

This data type is used to model media addresses. For many types of media, this will be in a binary representation. For example, an ethernet address would be represented as a string of 6 octets.

Groups in MIB-II

systemOBJECT IDENTIFIER ::= { mib-2 1 }

interfacesOBJECT IDENTIFIER ::= { mib-2 2 }

atOBJECT IDENTIFIER ::= { mib-2 3 }

ipOBJECT IDENTIFIER ::= { mib-2 4 }

icmpOBJECT IDENTIFIER ::= { mib-2 5 }

tcpOBJECT IDENTIFIER ::= { mib-2 6 }

udpOBJECT IDENTIFIER ::= { mib-2 7 }

egpOBJECT IDENTIFIER ::= { mib-2 8 }

Historical

cmotOBJECT IDENTIFIER ::= { mib-2 9 }

transmissionOBJECT IDENTIFIER ::= { mib-2 10 }

snmpOBJECT IDENTIFIER ::= { mib-2 11 }

System Group

Implementation of the system group is mandatory for all systems. If an agent is not configured to have a value for any of these variables, a string of length 0 is returned.

sysDescr OBJECT-TYPE

SYNTAX DisplayString (SIZE (0..255))

ACCESS read-only

STATUS mandatory

DESCRIPTION

A textual description of the entity. This value should include the full name and version identification of the system's hardware type, software operating-system, and networking software. It is mandatory that this only contain printable ASCII characters.

::= { system 1 }

sysObjectID OBJECT-TYPE

SYNTAX Object Identifier

ACCESS read-only

STATUS mandatory

DESCRIPTION

The vendor authoritative identification of the network management subsystem contained in the entity. This value is allocated within the SMI enterprises subtree (1.3.6.1.4.1) and provides an easy and unambiguous means for determining “what kind of box” is being managed. For example, if vendor “Flintstones, Inc.” was assigned the subtree 1.3.6.1.4.1.4242, it could assign the identifier 1.3.6.1.4.1.4242.1.1 to its “Fred Router”.

::= { system 2 }

sysUpTime OBJECT-TYPE

SYNTAX TimeTicks

ACCESS read-only

STATUS mandatory

DESCRIPTION

The time (in hundredths of a second) since the network management portion of the system was last re-initialized.

::= { system 3 }

sysContact OBJECT-TYPE

SYNTAX DisplayString (SIZE (0..255))

ACCESS read-write

STATUS mandatory

DESCRIPTION

The textual identification of the contact person for this managed node, together with information on how to contact this person.

::= { system 4 }

sysName OBJECT-TYPE

SYNTAX DisplayString (SIZE (0..255))

ACCESS read-write

STATUS mandatory

DESCRIPTION

An administratively-assigned name for this managed node. By convention, this is the node's fully-qualified domain name.

::= { system 5 }

sysLocation OBJECT-TYPE

SYNTAX DisplayString (SIZE (0..255))

ACCESS read-write

STATUS mandatory

DESCRIPTION

The physical location of this node (e.g., telephone closet, 3rd floor).

::= { system 6 }

sysServices OBJECT-TYPE

SYNTAX Integer (0..127)

ACCESS read-only

STATUS mandatory

DESCRIPTION

A value which indicates the set of services that this entity primarily offers. The value is a sum. This sum initially takes the value zero, Then, for each layer, L, in the range 1 through 7, that this node performs transactions for, 2 raised to (L - 1) is added to the sum. For example, a node which performs primarily routing functions would have a value of 4 (2^(3-1)). In contrast, a node which is a host offering application services would have a value of 72 (2^(4-1) + 2^(7-1)). Note that in the context of the Internet suite of protocols, values should be calculated accordingly (layer first, then functionality):

1 physical (e.g., repeaters)

2 datalink/subnetwork (e.g., bridges)

3 internet (e.g., IP gateways)

4 end-to-end (e.g., IP hosts)

7 applications (e.g., mail relays)

For systems including OSI protocols, layers 5 and 6 may also be counted.

::= { system 7 }

Interfaces Group

Implementation of the Interfaces group is mandatory for all systems.

ifNumber OBJECT-TYPE

SYNTAX Integer

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of network interfaces (regardless of their current state) present on this system.

::= { interfaces 1 }

Interfaces Table

The interfaces table contains information on the entity interfaces. Each interface is thought of as being attached to a subnetwork. Note that this term should not be confused with subnet which refers to an addressing partitioning scheme used in the Internet suite of protocols.

ifTable OBJECT-TYPE

SYNTAX Sequence of ifEntry

ACCESS not-accessible

STATUS mandatory

DESCRIPTION

A list of interface entries. The number of entries is given by the value of ifNumber.

::= { interfaces 2 }

ifEntry OBJECT-TYPE

SYNTAX IfEntry

ACCESS not-accessible

STATUS mandatory

DESCRIPTION

An interface entry containing objects at the subnetwork layer and below for a particular interface.

INDEX { ifIndex }

::= { ifTable 1 }

IfEntry ::=

SEQUENCE { ifIndex INTEGER, ifDescr DisplayString, ifType IANAifType, ifMtu INTEGER, ifSpeed Gauge, ifPhysAddress PhysAddress, ifAdminStatus INTEGER, ifOperStatus INTEGER, ifLastChange TimeTicks, ifInOctets Counter, ifInUcastPkts Counter, ifInNUcastPkts Counter, ifInDiscards Counter, ifInErrors Counter, ifInUnknownProtos Counter, ifOutOctets Counter, ifOutUcastPkts Counter, ifOutNUcastPkts Counter, ifOutDiscards Counter, ifOutErrors Counter, ifOutQLen Gauge, ifSpecific OBJECT IDENTIFIER }

ifIndex OBJECT-TYPE

SYNTAX Integer

ACCESS read-only

STATUS mandatory

DESCRIPTION

A unique value for each interface. Its value ranges between 1 and the value of ifNumber. The value for each interface must remain constant at least from one re-initialization of the entity network management system to the next re- initialization.

::= { ifEntry 1 }

ifDescr OBJECT-TYPE

SYNTAX DisplayString (SIZE (0..255))

ACCESS read-only

STATUS mandatory

DESCRIPTION

A textual string containing information about the interface. This string should include the name of the manufacturer, the product name and the version of the hardware interface.

::= { ifEntry 2 }

ifType OBJECT-TYPE

SYNTAX IANAifType

ACCESS read-only

STATUS mandatory

DESCRIPTION

The type of interface. Additional values for ifType are assigned by the Internet Assigned Numbers Authority (IANA), through updating the syntax of the IANAifType textual convention.

::= { ifEntry 3 }

ifMtu OBJECT-TYPE

SYNTAX Integer

ACCESS read-only

STATUS mandatory

DESCRIPTION

The size of the largest datagram which can be sent/received on the interface, specified in octets. For interfaces that are used for transmitting network datagrams, this is the size of the largest network datagram that can be sent on the interface.

::= { ifEntry 4 }

ifSpeed OBJECT-TYPE

SYNTAX Gauge

ACCESS read-only

STATUS mandatory

DESCRIPTION

An estimate of the interface current bandwidth in bits per second. For interfaces which do not vary in bandwidth or for those where no accurate estimation can be made, this object should contain the nominal bandwidth.

::= { ifEntry 5 }

ifPhysAddress OBJECT-TYPE

SYNTAX PhysAddress

ACCESS read-only

STATUS mandatory

DESCRIPTION

The interface address at the protocol layer immediately below the network layer in the protocol stack. For interfaces which do not have such an address (e.g., a serial line), this object should contain an octet string of zero length.

::= { ifEntry 6 }

ifAdminStatus OBJECT-TYPE

SYNTAX Integer { up(1), ready to pass packets down(2), testing(3) in some test mode }

ACCESS read-write

STATUS mandatory

DESCRIPTION

The desired state of the interface. The testing(3) state indicates that no operational packets can be passed.

::= { ifEntry 7 }

ifOperStatus OBJECT-TYPE

SYNTAX INTEGER { up(1), -- ready to pass packets down(2), testing(3), -- in some test mode

unknown(4), dormant(5) }

ACCESS read-only

STATUS mandatory

DESCRIPTION

The current operational state of the interface. The testing(3) state indicates that no operational packets can be passed.

::= { ifEntry 8 }

ifLastChange OBJECT-TYPE

SYNTAX TimeTicks

ACCESS read-only

STATUS mandatory

DESCRIPTION

The value of sysUpTime at the time the interface entered its current operational state. If the current state was entered prior to the last re- initialization of the local network management subsystem, then this object contains a zero value.

::= { ifEntry 9 }

ifInOctets OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION The total number of octets received on the interface, including framing characters.

::= { ifEntry 10 }

ifInUcastPkts OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of subnetwork-unicast packets delivered to a higher-layer protocol.

::= { ifEntry 11 }

ifInNUcastPkts OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of non-unicast (i.e., subnetwork- broadcast or subnetwork-multicast) packets delivered to a higher-layer protocol.

::= { ifEntry 12 }

ifInDiscards OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. One possible reason for discarding such a packet could be to free up buffer space.

::= { ifEntry 13 }

ifInErrors OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol.

::= { ifEntry 14 }

ifInUnknownProtos OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of packets received via the interface which were discarded because of an unknown or unsupported protocol.

::= { ifEntry 15 }

ifOutOctets OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of octets transmitted out of the interface, including framing characters.

::= { ifEntry 16 }

ifOutUcastPkts OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of packets that higher-level protocols requested be transmitted to a subnetwork-unicast address, including those that were discarded or not sent.

::= { ifEntry 17 }

ifOutNUcastPkts OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of packets that higher-level protocols requested be transmitted to a non- unicast (i.e., a subnetwork-broadcast or subnetwork-multicast) address, including those that were discarded or not sent.

::= { ifEntry 18 }

ifOutDiscards OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space.

::= { ifEntry 19 }

ifOutErrors OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of outbound packets that could not be transmitted because of errors.

::= { ifEntry 20 }

ifOutQLen OBJECT-TYPE

SYNTAX Gauge

ACCESS read-only

STATUS mandatory

DESCRIPTION

The length of the output packet queue (in packets).

::= { ifEntry 21 }

ifSpecific OBJECT-TYPE

SYNTAX OBJECT IDENTIFIER

ACCESS read-only

STATUS mandatory

DESCRIPTION

A reference to MIB definitions specific to the particular media being used to realize the interface. For example, if the interface is realized by an ethernet, then the value of this object refers to a document defining objects specific to ethernet. If this information is not present, its value should be set to the OBJECT IDENTIFIER { 0 0 }, which is a syntactically valid object identifier, and any conformant implementation of ASN.1 and BER must be able to generate and recognize this value.

::= { ifEntry 22 }

Address Translation Group

Implementation of the Address Translation group is mandatory for all systems. Note however that this group is deprecated by MIB-II. That is, it is being included solely for compatibility with MIB-I nodes, and will most likely be excluded from MIB-III nodes. From MIB-II and onwards, each network protocol group contains its own address translation tables. The Address Translation group contains one table which is the union across all interfaces of the translation tables for converting a NetworkAddress (e.g., an IP address) into a subnetwork-specific address. For lack of a better term, this document refers to such a subnetwork-specific address as a physical address.

Examples of such translation tables are: for broadcast media where ARP is in use, the translation table is equivalent to the ARP cache; or, on an X.25 network where non-algorithmic translation to X.121 addresses is required, the translation table contains the NetworkAddress to X.121 address equivalences.

atTable OBJECT-TYPE

SYNTAX Sequence of atEntry

ACCESS not-accessible

STATUS deprecated

DESCRIPTION

The Address Translation tables contain the NetworkAddress to physical address equivalences. Some interfaces do not use translation tables for determining address equivalences (e.g., DDN-X.25 has an algorithmic method); if all interfaces are of this type, then the Address Translation table is empty, i.e., has zero entries.

::= { at 1 }

atEntry OBJECT-TYPE

SYNTAX AtEntry

ACCESS not-accessible

STATUS deprecated

DESCRIPTION

Each entry contains one NetworkAddress to physical address equivalence.

INDEX { atIfIndex, atNetAddress }

::= { atTable 1 }

AtEntry ::=

SEQUENCE { atIfIndex INTEGER, atPhysAddress PhysAddress, atNetAddress NetworkAddress }

atIfIndex OBJECT-TYPE

SYNTAX Integer

ACCESS read-write

STATUS deprecated

DESCRIPTION

The interface on which this entry equivalence is effective. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex.

::= { atEntry 1 }

atPhysAddress OBJECT-TYPE

SYNTAX PhysAddress

ACCESS read-write

STATUS deprecated

DESCRIPTION

The media-dependent physical address. Setting this object to a null string (one of zero length) has the effect of invaliding the corresponding entry in the atTable object. That is, it effectively disassociates the interface identified with said entry from the mapping identified with said entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table.

Accordingly, management stations must be prepared to receive tabular information from agents that corresponds to entries not currently in use.

Proper interpretation of such entries requires examination of the relevant atPhysAddress object.

::= { atEntry 2 }

atNetAddress OBJECT-TYPE

SYNTAX NetworkAddress

ACCESS read-write

STATUS deprecated

DESCRIPTION

The NetworkAddress (e.g., the IP address) corresponding to the media-dependent physical address.

::= { atEntry 3 }

IP Group

Implementation of the IP group is mandatory for all systems.

ipForwarding OBJECT-TYPE

SYNTAX INTEGER { forwarding(1), -- acting as a gateway not-forwarding(2) -- NOT acting as a gateway }

ACCESS read-write

STATUS mandatory

DESCRIPTION

The indication of whether this entity is acting as an IP gateway in respect to the forwarding of datagrams received by, but not addressed to, this entity. IP gateways forward datagrams. IP hosts do not (except those source-routed via the host). Note that for some managed nodes, this object may take on only a subset of the values possible. Accordingly, it is appropriate for an agent to return a badValue response if a management station attempts to change this object to an inappropriate value.

::= { ip 1 }

ipDefaultTTL OBJECT-TYPE

SYNTAX Integer

ACCESS read-write

STATUS mandatory

DESCRIPTION

The default value inserted into the Time-To-Live field of the IP header of datagrams originated at this entity, whenever a TTL value is not supplied by the transport layer protocol.

::= { ip 2 }

ipInReceives OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of input datagrams received from interfaces, including those received in error.

::= { ip 3 }

ipInHdrErrors OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of input datagrams discarded due to errors in their IP headers, including bad checksums, version number mismatch, other format errors, time-to-live exceeded, errors discovered in processing their IP options, etc.

::= { ip 4 }

ipInAddrErrors OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of input datagrams discarded because the IP address in their IP header's destination field was not a valid address to be received at this entity. This count includes invalid addresses (e.g., 0.0.0.0) and addresses of unsupported Classes (e.g., Class E). For entities which are not IP Gateways and therefore do not forward datagrams, this counter includes datagrams discarded because the destination address was not a local address.

::= { ip 5 }

ipForwDatagrams OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of input datagrams for which this entity was not their final IP destination, as a result of which an attempt was made to find a route to forward them to that final destination. In entities which do not act as IP Gateways, this counter will include only those packets which were Source-Routed via this entity, and the Source- Route option processing was successful.

::= { ip 6 }

ipInUnknownProtos OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of locally-addressed datagrams received successfully but discarded because of an unknown or unsupported protocol.

::= { ip 7 }

ipInDiscards OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of input IP datagrams for which no problems were encountered to prevent their continued processing, but which were discarded (e.g., for lack of buffer space). Note that this counter does not include any datagrams discarded while awaiting re-assembly.

::= { ip 8 }

ipInDelivers OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of input datagrams successfully delivered to IP user-protocols (including ICMP).

::= { ip 9 }

ipOutRequests OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of IP datagrams which local IP user-protocols (including ICMP) supplied to IP in requests for transmission. Note that this counter does not include any datagrams counted in ipForwDatagrams.

::= { ip 10 }

ipOutDiscards OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of output IP datagrams for which no problem was encountered to prevent their transmission to their destination, but which were discarded (e.g., for lack of buffer space). Note that this counter would include datagrams counted in ipForwDatagrams if any such packets met this (discretionary) discard criterion.

::= { ip 11 }

ipOutNoRoutes OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of IP datagrams discarded because no route could be found to transmit them to their destination. Note that this counter includes any packets counted in ipForwDatagrams which meet this no-route criterion. Note that this includes any datagrams which a host cannot route because all of its default gateways are down.

::= { ip 12 }

ipReasmTimeout OBJECT-TYPE

SYNTAX Integer

ACCESS read-only

STATUS mandatory

DESCRIPTION

The maximum number of seconds which received fragments are held while they are awaiting reassembly at this entity.

::= { ip 13 }

ipReasmReqds OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of IP fragments received which needed to be reassembled at this entity.

::= { ip 14 }

ipReasmOKs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of IP datagrams successfully re-assembled.

::= { ip 15 }

ipReasmFails OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of failures detected by the IP re-assembly algorithm (for whatever reason: timed out, errors, etc). Note that this is not necessarily a count of discarded IP fragments since some algorithms (notably the algorithm in RFC 815) can lose track of the number of fragments by combining them as they are received.

::= { ip 16 }

ipFragOKs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of IP datagrams that have been successfully fragmented at this entity.

::= { ip 17 }

ipFragFails OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of IP datagrams that have been discarded because they needed to be fragmented at this entity but could not be, e.g., because their Don't Fragment flag was set.

::= { ip 18 }

ipFragCreates OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of IP datagram fragments that have been generated as a result of fragmentation at this entity.

::= { ip 19 }

IP Address Table

The IP address table contains this entity IP addressing information.

ipAddrTable OBJECT-TYPE

SYNTAX Sequence of ipAddrEntry

ACCESS not-accessible

STATUS mandatory

DESCRIPTION

The table of addressing information relevant to this entity IP addresses.

::= { ip 20 }

ipAddrEntry OBJECT-TYPE

SYNTAX IpAddrEntry

ACCESS not-accessible

STATUS mandatory

DESCRIPTION

The addressing information for one of this entity IP addresses.

INDEX { ipAdEntAddr }

::= { ipAddrTable 1 }

IpAddrEntry ::=

SEQUENCE { ipAdEntAddr IpAddress, ipAdEntIfIndex INTEGER, ipAdEntNetMask IpAddress, ipAdEntBcastAddr INTEGER, ipAdEntReasmMaxSize INTEGER (0..65535) }

ipAdEntAddr OBJECT-TYPE

SYNTAX IpAddress

ACCESS read-only

STATUS mandatory

DESCRIPTION

The IP address to which this entry addressing information pertains.

::= { ipAddrEntry 1 }

ipAdEntIfIndex OBJECT-TYPE

SYNTAX Integer

ACCESS read-only

STATUS mandatory

DESCRIPTION

The index value which uniquely identifies the interface to which this entry is applicable. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex.

::= { ipAddrEntry 2 }

ipAdEntNetMask OBJECT-TYPE

SYNTAX IpAddress

ACCESS read-only

STATUS mandatory

DESCRIPTION

The subnet mask associated with the IP address of this entry. The value of the mask is an IP address with all the network bits set to 1 and all the hosts bits set to 0.

::= { ipAddrEntry 3 }

ipAdEntBcastAddr OBJECT-TYPE

SYNTAX Integer

ACCESS read-only

STATUS mandatory

DESCRIPTION

The value of the least-significant bit in the IP broadcast address used for sending datagrams on the (logical) interface associated with the IP address of this entry. For example, when the Internet standard all-ones broadcast address is used, the value will be 1. This value applies to both the subnet and network broadcasts addresses used by the entity on this (logical) interface.

::= { ipAddrEntry 4 }

ipAdEntReasmMaxSize OBJECT-TYPE

SYNTAX Integer (0..65535)

ACCESS read-only

STATUS mandatory

DESCRIPTION

The size of the largest IP datagram which this entity can re-assemble from incoming IP fragmented datagrams received on this interface.

::= { ipAddrEntry 5 }

IP Routing Table

-- The IP routing table contains an entry for each route

-- presently known to this entity.

ipRouteTable OBJECT-TYPE

SYNTAX Sequence of ipRouteEntry

ACCESS not-accessible

STATUS mandatory

DESCRIPTION

This entity IP Routing table.

::= { ip 21 }

ipRouteEntry OBJECT-TYPE

SYNTAX IpRouteEntry

ACCESS not-accessible

STATUS mandatory

DESCRIPTION

A route to a particular destination.

INDEX { ipRouteDest }

::= { ipRouteTable 1 }

IpRouteEntry ::=

SEQUENCE { ipRouteDest IpAddress, ipRouteIfIndex INTEGER, ipRouteMetric1 INTEGER, ipRouteMetric2 INTEGER, ipRouteMetric3 INTEGER, ipRouteMetric4 INTEGER, ipRouteNextHop IpAddress, ipRouteType INTEGER, ipRouteProto INTEGER, ipRouteAge INTEGER, ipRouteMask IpAddress, ipRouteMetric5 INTEGER, ipRouteInfo OBJECT IDENTIFIER }

ipRouteDest OBJECT-TYPE

SYNTAX IpAddress

ACCESS read-write

STATUS mandatory

DESCRIPTION

The destination IP address of this route. An entry with a value of 0.0.0.0 is considered a default route. Multiple routes to a single destination can appear in the table, but access to such multiple entries is dependent on the table-access mechanisms defined by the network management protocol in use.

::= { ipRouteEntry 1 }

ipRouteIfIndex OBJECT-TYPE

SYNTAX Integer

ACCESS read-write

STATUS mandatory

DESCRIPTION

The index value which uniquely identifies the local interface through which the next hop of this route should be reached. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex.

::= { ipRouteEntry 2 }

ipRouteMetric1 OBJECT-TYPE

SYNTAX Integer

ACCESS read-write

STATUS mandatory

DESCRIPTION

The primary routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route ipRouteProto value. If this metric is not used, its value should be set to -1.

::= { ipRouteEntry 3 }

ipRouteMetric2 OBJECT-TYPE

SYNTAX Integer

ACCESS read-write

STATUS mandatory

DESCRIPTION

An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route ipRouteProto value. If this metric is not used, its value should be set to -1.

::= { ipRouteEntry 4 }

ipRouteMetric3 OBJECT-TYPE

SYNTAX Integer

ACCESS read-write

STATUS mandatory

DESCRIPTION

An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route ipRouteProto value. If this metric is not used, its value should be set to -1.

::= { ipRouteEntry 5 }

ipRouteMetric4 OBJECT-TYPE

SYNTAX Integer

ACCESS read-write

STATUS mandatory

DESCRIPTION

An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route ipRouteProto value. If this metric is not used, its value should be set to -1.

::= { ipRouteEntry 6 }

ipRouteNextHop OBJECT-TYPE

SYNTAX IpAddress

ACCESS read-write

STATUS mandatory

DESCRIPTION

The IP address of the next hop of this route. (In the case of a route bound to an interface which is realized via a broadcast media, the value of this field is the agent's IP address on that interface.)

::= { ipRouteEntry 7 }

ipRouteType OBJECT-TYPE

SYNTAX Integer { other(1), -- none of the following invalid(2), -- an invalidated route -- route to directly direct(3), -- connected (sub-)network -- route to a non-local indirect(4) -- host/network/sub-network }

ACCESS read-write

STATUS mandatory

DESCRIPTION

The type of route. Note that the values direct(3) and indirect(4) refer to the notion of direct and indirect routing in the IP architecture. Setting this object to the value invalid(2) has the effect of invalidating the corresponding entry in the ipRouteTable object. That is, it effectively disassociates the destination identified with said entry from the route identified with said entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table.

Accordingly, management stations must be prepared to receive tabular information from agents that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant ipRouteType object.

::= { ipRouteEntry 8 }

ipRouteProto OBJECT-TYPE

SYNTAX INTEGER { other(1), -- none of the following -- non-protocol information, -- e.g., manually configured local(2), -- entries -- set via a network netmgmt(3), -- management protocol -- obtained via ICMP, icmp(4), -- e.g., Redirect -- the remaining values are -- all gateway routing -- protocols egp(5), ggp(6), hello(7), rip(8), is-is(9), es-is(10), ciscoIgrp(11), bbnSpfIgp(12), ospf(13), bgp(14) }

ACCESS read-only

STATUS mandatory

DESCRIPTION

The routing mechanism via which this route was learned. Inclusion of values for gateway routing protocols is not intended to imply that hosts should support those protocols.

::= { ipRouteEntry 9 }

ipRouteAge OBJECT-TYPE

SYNTAX Integer

ACCESS read-write

STATUS mandatory

DESCRIPTION

The number of seconds since this route was last updated or otherwise determined to be correct. Note that no semantics of too old can be implied except through knowledge of the routing protocol by which the route was learned.

::= { ipRouteEntry 10 }

ipRouteMask OBJECT-TYPE

SYNTAX IpAddress

ACCESS read-write

STATUS mandatory

DESCRIPTION

Indicate the mask to be logical-ANDed with the destination address before being compared to the value in the ipRouteDest field. For those systems that do not support arbitrary subnet masks, an agent constructs the value of the ipRouteMask by determining whether the value of the correspondent ipRouteDest field belong to a class-A, B, or C network, and then using one of: mask network 255.0.0.0 class-A, 255.255.0.0 class-B, 255.255.255.0 class-C. If the value of the ipRouteDest is 0.0.0.0 (a default route), then the mask value is also 0.0.0.0. It should be noted that all IP routing subsystems implicitly use this mechanism.

::= { ipRouteEntry 11 }

ipRouteMetric5 OBJECT-TYPE

SYNTAX Integer

ACCESS read-write

STATUS mandatory

DESCRIPTION

An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route ipRouteProto value. If this metric is not used, its value should be set to -1.

::= { ipRouteEntry 12 }

ipRouteInfo OBJECT-TYPE

SYNTAX OBJECT IDENTIFIER

ACCESS read-only

STATUS mandatory

DESCRIPTION

A reference to MIB definitions specific to the particular routing protocol which is responsible for this route, as determined by the value specified in the route ipRouteProto value. If this information is not present, its value should be set to the OBJECT IDENTIFIER { 0 0 }, which is a syntactically valid object identifier, and any conformant implementation of ASN.1 and BER must be able to generate and recognize this value.

::= { ipRouteEntry 13 }

IP Address Translation Table

The IP address translation table contain the IP Address to physical address equivalences. Some interfaces do not use translation tables for determining address equivalences (e.g., DDN-X.25 has an algorithmic method); if all interfaces are of this type, then the Address Translation table is empty, i.e., has zero entries.

ipNetToMediaTable OBJECT-TYPE

SYNTAX Sequence of ipNetToMediaEntry

ACCESS not-accessible

STATUS mandatory

DESCRIPTION

The IP Address Translation table used for mapping from IP addresses to physical addresses.

::= { ip 22 }

ipNetToMediaEntry OBJECT-TYPE

SYNTAX IpNetToMediaEntry

ACCESS not-accessible

STATUS mandatory

DESCRIPTION

Each entry contains one IpAddress to physical address equivalence.

INDEX { ipNetToMediaIfIndex, ipNetToMediaNetAddress }

::= { ipNetToMediaTable 1 }

IpNetToMediaEntry ::=

SEQUENCE { ipNetToMediaIfIndex INTEGER, ipNetToMediaPhysAddress PhysAddress, ipNetToMediaNetAddress IpAddress, ipNetToMediaType INTEGER }

ipNetToMediaIfIndex OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS mandatory

DESCRIPTION

The interface on which this entry's equivalence is effective. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex.

::= { ipNetToMediaEntry 1 }

ipNetToMediaPhysAddress OBJECT-TYPE

SYNTAX PhysAddress

ACCESS read-write

STATUS mandatory

DESCRIPTION

The media-dependent physical address.

::= { ipNetToMediaEntry 2 }

ipNetToMediaNetAddress OBJECT-TYPE

SYNTAX IpAddress

ACCESS read-write

STATUS mandatory

DESCRIPTION

The IpAddress corresponding to the media- dependent physical address.

::= { ipNetToMediaEntry 3 }

ipNetToMediaType OBJECT-TYPE

SYNTAX Integer { other(1), -- none of the following invalid(2), -- an invalidated mapping dynamic(3), static(4) }

ACCESS read-write

STATUS mandatory

DESCRIPTION

The type of mapping. Setting this object to the value invalid(2) has the effect of invalidating the corresponding entry in the ipNetToMediaTable. That is, it effectively disassociates the interface identified with said entry from the mapping identified with said entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management stations must be prepared to receive tabular information from agents that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant ipNetToMediaType object.

::= { ipNetToMediaEntry 4 }

Additional IP Objects

ipRoutingDiscards OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of routing entries which were chosen to be discarded even though they are valid. One possible reason for discarding such an entry could be to free-up buffer space for other routing entries.

::= { ip 23 }

ICMP Group

Implementation of the ICMP group is mandatory for all systems.

icmpInMsgs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of ICMP messages which the entity received. Note that this counter includes all those counted by icmpInErrors.

::= { icmp 1 }

icmpInErrors OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of ICMP messages which the entity received but determined as having ICMP-specific errors (bad ICMP checksums, bad length, etc.).

::= { icmp 2 }

icmpInDestUnreachs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of ICMP Destination Unreachable messages received.

::= { icmp 3 }

icmpInTimeExcds OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of ICMP Time Exceeded messages received.

::= { icmp 4 }

icmpInParmProbs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of ICMP Parameter Problem messages received.

::= { icmp 5 }

icmpInSrcQuenchs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of ICMP Source Quench messages received.

::= { icmp 6 }

icmpInRedirects OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of ICMP Redirect messages received.

::= { icmp 7 }

icmpInEchos OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of ICMP Echo (request) messages received.

::= { icmp 8 }

icmpInEchoReps OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of ICMP Echo Reply messages received.

::= { icmp 9 }

icmpInTimestamps OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of ICMP Timestamp (request) messages received.

::= { icmp 10 }

icmpInTimestampReps OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of ICMP Timestamp Reply messages received.

::= { icmp 11 }

icmpInAddrMasks OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of ICMP Address Mask Request messages received.

::= { icmp 12 }

icmpInAddrMaskReps OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of ICMP Address Mask Reply messages received.

::= { icmp 13 }

icmpOutMsgs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of ICMP messages which this entity attempted to send. Note that this counter includes all those counted by icmpOutErrors.

::= { icmp 14 }

icmpOutErrors OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of ICMP messages which this entity did not send due to problems discovered within ICMP such as a lack of buffers. This value should not include errors discovered outside the ICMP layer such as the inability of IP to route the resultant datagram. In some implementations there may be no types of error which contribute to this counter value.

::= { icmp 15 }

icmpOutDestUnreachs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of ICMP Destination Unreachable messages sent.

::= { icmp 16 }

icmpOutTimeExcds OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of ICMP Time Exceeded messages sent.

::= { icmp 17 }

icmpOutParmProbs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of ICMP Parameter Problem messages

sent.

::= { icmp 18 }

icmpOutSrcQuenchs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of ICMP Source Quench messages sent.

::= { icmp 19 }

icmpOutRedirects OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of ICMP Redirect messages sent. For a host, this object will always be zero, since hosts do not send redirects.

::= { icmp 20 }

icmpOutEchos OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of ICMP Echo (request) messages sent.

::= { icmp 21 }

icmpOutEchoReps OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of ICMP Echo Reply messages sent.

::= { icmp 22 }

icmpOutTimestamps OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of ICMP Timestamp (request) messages sent.

::= { icmp 23 }

icmpOutTimestampReps OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of ICMP Timestamp Reply messages sent.

::= { icmp 24 }

icmpOutAddrMasks OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of ICMP Address Mask Request messages sent.

::= { icmp 25 }

icmpOutAddrMaskReps OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of ICMP Address Mask Reply messages sent.

::= { icmp 26 }

TCP Group

Implementation of the TCP group is mandatory for all systems that implement the TCP. Note that instances of object types that represent information about a particular TCP connection are transient; they persist only as long as the connection in question.

tcpRtoAlgorithm OBJECT-TYPE

SYNTAX Integer { other(1), -- none of the following constant(2), -- a constant rto rsre(3), -- MIL-STD-1778, Appendix B vanj(4) -- Van Jacobson's algorithm [10] }

ACCESS read-only

STATUS mandatory

DESCRIPTION

The algorithm used to determine the timeout value used for retransmitting unacknowledged octets.

::= { tcp 1 }

tcpRtoMin OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-only

STATUS mandatory

DESCRIPTION

The minimum value permitted by a TCP implementation for the retransmission timeout, measured in milliseconds. More refined semantics for objects of this type depend upon the algorithm used to determine the retransmission timeout. In particular, when the timeout algorithm is rsre(3), an object of this type has the semantics of the LBOUND quantity described in RFC 793.

::= { tcp 2 }

tcpRtoMax OBJECT-TYPE

SYNTAX Integer

ACCESS read-only

STATUS mandatory

DESCRIPTION

The maximum value permitted by a TCP implementation for the retransmission timeout, measured in milliseconds. More refined semantics for objects of this type depend upon the algorithm used to determine the retransmission timeout. In particular, when the timeout algorithm is rsre(3), an object of this type has the semantics of the UBOUND quantity described in RFC 793.

::= { tcp 3 }

tcpMaxConn OBJECT-TYPE

SYNTAX Integer

ACCESS read-only

STATUS mandatory

DESCRIPTION

The limit on the total number of TCP connections the entity can support. In entities where the maximum number of connections is dynamic, this object should contain the value -1.

::= { tcp 4 }

tcpActiveOpens OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of times TCP connections have made a direct transition to the SYN-SENT state from the CLOSED state.

::= { tcp 5 }

tcpPassiveOpens OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of times TCP connections have made a direct transition to the SYN-RCVD state from the LISTEN state.

::= { tcp 6 }

tcpAttemptFails OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of times TCP connections have made a direct transition to the CLOSED state from either the SYN-SENT state or the SYN-RCVD state, plus the number of times TCP connections have made a direct transition to the LISTEN state from the SYN-RCVD state.

::= { tcp 7 }

tcpEstabResets OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of times TCP connections have made a direct transition to the CLOSED state from either the ESTABLISHED state or the CLOSE-WAIT state.

::= { tcp 8 }

tcpCurrEstab OBJECT-TYPE

SYNTAX Gauge

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of TCP connections for which the current state is either ESTABLISHED or CLOSE- WAIT.

::= { tcp 9 }

tcpInSegs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of segments received, including those received in error. This count includes segments received on currently established connections.

::= { tcp 10 }

tcpOutSegs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of segments sent, including those on current connections but excluding those containing only retransmitted octets.

::= { tcp 11 }

tcpRetransSegs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of segments retransmitted that is, the number of TCP segments transmitted containing one or more previously transmitted octets.

::= { tcp 12 }

TCP Connection Table

The TCP connection table contains information about this entity existing TCP connections.

tcpConnTable OBJECT-TYPE

SYNTAX Sequence of tcpConnEntry

ACCESS not-accessible

STATUS mandatory

DESCRIPTION

A table containing TCP connection-specific information.

::= { tcp 13 }

tcpConnEntry OBJECT-TYPE

SYNTAX TcpConnEntry

ACCESS not-accessible

STATUS mandatory

DESCRIPTION

Information about a particular current TCP connection. An object of this type is transient, in that it ceases to exist when (or soon after) the connection makes the transition to the CLOSED state.

INDEX { tcpConnLocalAddress, tcpConnLocalPort, tcpConnRemAddress, tcpConnRemPort }

::= { tcpConnTable 1 }

TcpConnEntry ::=

SEQUENCE { tcpConnState INTEGER, tcpConnLocalAddress IpAddress, tcpConnLocalPort INTEGER (0..65535), tcpConnRemAddress IpAddress, tcpConnRemPort INTEGER (0..65535) }

tcpConnState OBJECT-TYPE

SYNTAX INTEGER { closed(1), listen(2), synSent(3), synReceived(4), established(5), finWait1(6), finWait2(7), closeWait(8), lastAck(9), closing(10), timeWait(11), deleteTCB(12) }

ACCESS read-write

STATUS mandatory

DESCRIPTION

The state of this TCP connection. The only value which may be set by a management station is deleteTCB(12). Accordingly, it is appropriate for an agent to return a badValue response if a management station attempts to set this object to any other value. If a management station sets this object to the value

deleteTCB(12), then this has the effect of deleting the TCB (as defined in RFC 793) of the corresponding connection on the managed node, resulting in immediate termination of the connection.

As an implementation-specific option, a RST segment may be sent from the managed node to the other TCP endpoint (note however that RST segments are not sent reliably).

::= { tcpConnEntry 1 }

tcpConnLocalAddress OBJECT-TYPE

SYNTAX IpAddress

ACCESS read-only

STATUS mandatory

DESCRIPTION

The local IP address for this TCP connection. In the case of a connection in the listen state which is willing to accept connections for any IP interface associated with the node, the value 0.0.0.0 is used.

::= { tcpConnEntry 2 }

tcpConnLocalPort OBJECT-TYPE

SYNTAX INTEGER (0..65535)

ACCESS read-only

STATUS mandatory

DESCRIPTION

The local port number for this TCP connection.

::= { tcpConnEntry 3 }

tcpConnRemAddress OBJECT-TYPE

SYNTAX IpAddress

ACCESS read-only

STATUS mandatory

DESCRIPTION

The remote IP address for this TCP connection.

::= { tcpConnEntry 4 }

tcpConnRemPort OBJECT-TYPE

SYNTAX INTEGER (0..65535)

ACCESS read-only

STATUS mandatory

DESCRIPTION

The remote port number for this TCP connection.

::= { tcpConnEntry 5 }

Additional TCP Objects

tcpInErrs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of segments received in error (e.g., bad TCP checksums).

::= { tcp 14 }

tcpOutRsts OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of TCP segments sent containing the RST flag.

::= { tcp 15 }

UDP Group

Implementation of the UDP group is mandatory for all systems which implement the UDP.

udpInDatagrams OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of UDP datagrams delivered to UDP users.

::= { udp 1 }

udpNoPorts OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of received UDP datagrams for which there was no application at the destination port.

::= { udp 2 }

udpInErrors OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

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 Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of UDP datagrams sent from this entity.

::= { udp 4 }

UDP Listener Table

The UDP listener table contains information about this entity UDP end-points on which a local application is currently accepting datagrams.

udpTable OBJECT-TYPE

SYNTAX SEQUENCE OF UdpEntry

ACCESS not-accessible

STATUS mandatory

DESCRIPTION

A table containing UDP listener information.

::= { udp 5 }

udpEntry OBJECT-TYPE

SYNTAX UdpEntry

ACCESS not-accessible

STATUS mandatory

DESCRIPTION

Information about a particular current UDP listener.

INDEX { udpLocalAddress, udpLocalPort }

::= { udpTable 1 }

UdpEntry ::=

SEQUENCE { udpLocalAddress IpAddress, udpLocalPort INTEGER (0..65535) }

udpLocalAddress OBJECT-TYPE

SYNTAX IpAddress

ACCESS read-only

STATUS mandatory

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 INTEGER (0..65535)

ACCESS read-only

STATUS mandatory

DESCRIPTION

The local port number for this UDP listener.

::= { udpEntry 2 }

EGP Group

Implementation of the EGP group is mandatory for all systems which implement the EGP.

egpInMsgs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of EGP messages received without error.

::= { egp 1 }

egpInErrors OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of EGP messages received that proved to be in error.

::= { egp 2 }

egpOutMsgs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of locally generated EGP messages.

::= { egp 3 }

egpOutErrors OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of locally generated EGP messages not sent due to resource limitations within an EGP entity.

::= { egp 4 }

EGP Neighbor Table

The EGP neighbor table contains information about this entity EGP neighbors.

egpNeighTable OBJECT-TYPE

SYNTAX SEQUENCE OF EgpNeighEntry

ACCESS not-accessible

STATUS mandatory

DESCRIPTION

The EGP neighbor table.

::= { egp 5 }

egpNeighEntry OBJECT-TYPE

SYNTAX EgpNeighEntry

ACCESS not-accessible

STATUS mandatory

DESCRIPTION

Information about this entity's relationship with a particular EGP neighbor.

INDEX { egpNeighAddr }

::= { egpNeighTable 1 }

EgpNeighEntry ::=

SEQUENCE { egpNeighState INTEGER, egpNeighAddr IpAddress, egpNeighAs INTEGER, egpNeighInMsgs Counter, egpNeighInErrs Counter, egpNeighOutMsgs Counter, egpNeighOutErrs Counter, egpNeighInErrMsgs Counter, egpNeighOutErrMsgs Counter, egpNeighStateUps Counter, egpNeighStateDowns Counter, egpNeighIntervalHello INTEGER, egpNeighIntervalPoll INTEGER, egpNeighMode INTEGER, egpNeighEventTrigger INTEGER }

egpNeighState OBJECT-TYPE

SYNTAX Integer { idle(1), acquisition(2), down(3), up(4), cease(5) }

ACCESS read-only

STATUS mandatory

DESCRIPTION

The EGP state of the local system with respect to the entry EGP neighbor. Each EGP state is represented by a value that is one greater than the numerical value associated with said state in RFC 904.

::= { egpNeighEntry 1 }

egpNeighAddr OBJECT-TYPE

SYNTAX IpAddress

ACCESS read-only

STATUS mandatory

DESCRIPTION

The IP address of this entry's EGP neighbor.

::= { egpNeighEntry 2 }

egpNeighAs OBJECT-TYPE

SYNTAX Integer

ACCESS read-only

STATUS mandatory

DESCRIPTION

The autonomous system of this EGP peer. Zero should be specified if the autonomous system number of the neighbor is not yet known.

::= { egpNeighEntry 3 }

egpNeighInMsgs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of EGP messages received without error from this EGP peer.

::= { egpNeighEntry 4 }

egpNeighInErrs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of EGP messages received from this EGP peer that proved to be in error (e.g., bad EGP checksum).

::= { egpNeighEntry 5 }

egpNeighOutMsgs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of locally generated EGP messages to this EGP peer.

::= { egpNeighEntry 6 }

egpNeighOutErrs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of locally generated EGP messages not sent to this EGP peer due to resource limitations within an EGP entity.

::= { egpNeighEntry 7 }

egpNeighInErrMsgs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of EGP-defined error messages received from this EGP peer.

::= { egpNeighEntry 8 }

egpNeighOutErrMsgs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of EGP-defined error messages sent to this EGP peer.

::= { egpNeighEntry 9 }

egpNeighStateUps OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of EGP state transitions to the UP state with this EGP peer.

::= { egpNeighEntry 10 }

egpNeighStateDowns OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The number of EGP state transitions from the UP state to any other state with this EGP peer.

::= { egpNeighEntry 11 }

egpNeighIntervalHello OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-only

STATUS mandatory

DESCRIPTION

The interval between EGP Hello command retransmissions (in hundredths of a second). This represents the t1 timer as defined in RFC 904.

::= { egpNeighEntry 12 }

egpNeighIntervalPoll OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-only

STATUS mandatory

DESCRIPTION

The interval between EGP poll command retransmissions (in hundredths of a second). This represents the t3 timer as defined in RFC 904.

::= { egpNeighEntry 13 }

egpNeighMode OBJECT-TYPE

SYNTAX INTEGER { active(1), passive(2) }

ACCESS read-only

STATUS mandatory

DESCRIPTION

The polling mode of this EGP entity, either passive or active.

::= { egpNeighEntry 14 }

egpNeighEventTrigger OBJECT-TYPE

SYNTAX INTEGER { start(1), stop(2) }

ACCESS read-write

STATUS mandatory

DESCRIPTION

A control variable used to trigger operator-initiated Start and Stop events. When read, this variable always returns the most recent value that egpNeighEventTrigger was set to. If it has not been set since the last initialization of the network management subsystem on the node, it returns a value of stop. When set, this variable causes a Start or Stop event on the specified neighbor, as specified on pages 8-10 of RFC 904. Briefly, a Start event causes an Idle peer to begin neighbor acquisition and a non-Idle peer to reinitiate neighbor acquisition. A stop event causes a non-Idle peer to return to the Idle state until a Start event occurs, either via egpNeighEventTrigger or otherwise.

::= { egpNeighEntry 15 }

Additional EGP Objects

egpAs OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-only

STATUS mandatory

DESCRIPTION

The autonomous system number of this EGP entity.

::= { egp 6 }

Transmission Group

Based on the transmission media underlying each interface on a system, the corresponding portion of the Transmission group is mandatory for that system. When Internet-standard definitions for managing transmission media are defined, the transmission group is used to provide a prefix for the names of those objects. Typically, such definitions reside in the experimental portion of the MIB until they are proven, then as a part of the Internet standardization process, the definitions are accordingly elevated and a new object identifier, under the transmission group is defined. By convention, the name assigned is: type OBJECT IDENTIFIER ::= { transmission number } where type is the symbolic value used for the media in the ifType column of the ifTable object, and number is the actual integer value corresponding to the symbol.

SNMP Group

Implementation of the SNMP group is mandatory for all systems which support an SNMP protocol entity. Some of the objects defined below will be zero-valued in those SNMP implementations that are optimized to support only those functions specific to either a management agent or a management station. In particular, it should be observed that the objects below refer to an SNMP entity, and there may be several SNMP entities residing on a managed node (e.g., if the node is hosting acting as a management station).

snmpInPkts OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of Messages delivered to the SNMP entity from the transport service.

::= { snmp 1 }

snmpOutPkts OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of SNMP Messages which were passed from the SNMP protocol entity to the transport service.

::= { snmp 2 }

snmpInBadVersions OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of SNMP Messages which were delivered to the SNMP protocol entity and were for an unsupported SNMP version.

::= { snmp 3 }

snmpInBadCommunityNames OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of SNMP Messages delivered to the SNMP protocol entity which used a SNMP community name not known to said entity.

::= { snmp 4 }

snmpInBadCommunityUses OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of SNMP Messages delivered to the SNMP protocol entity which represented an SNMP operation which was not allowed by the SNMP community named in the Message.

::= { snmp 5 }

snmpInASNParseErrs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of ASN.1 or BER errors encountered by the SNMP protocol entity when decoding received SNMP Messages.

::= { snmp 6 }

-- { snmp 7 } is not used

snmpInTooBigs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of SNMP PDUs which were delivered to the SNMP protocol entity and for which the value of the error-status field is tooBig.

::= { snmp 8 }

snmpInNoSuchNames OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of SNMP PDUs which were delivered to the SNMP protocol entity and for which the value of the error-status field is noSuchName.

::= { snmp 9 }

snmpInBadValues OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of SNMP PDUs which were delivered to the SNMP protocol entity and for which the value of the error-status field is 'badValue'.

::= { snmp 10 }

snmpInReadOnlys OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number valid SNMP PDUs which were delivered to the SNMP protocol entity and for which the value of the error-status field is readOnly. It should be noted that it is a protocol error to generate an SNMP PDU which contains the value readOnly in the error-status field, as such this object is provided as a means of detecting incorrect implementations of the SNMP.

::= { snmp 11 }

snmpInGenErrs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of SNMP PDUs which were delivered to the SNMP protocol entity and for which the value of the error-status field is genErr.

::= { snmp 12 }

snmpInTotalReqVars OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of MIB objects which have been retrieved successfully by the SNMP protocol entity as the result of receiving valid SNMP Get-Request and Get-Next PDUs.

::= { snmp 13 }

snmpInTotalSetVars OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of MIB objects which have been altered successfully by the SNMP protocol entity as the result of receiving valid SNMP Set-Request PDUs.

::= { snmp 14 }

snmpInGetRequests OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of SNMP Get-Request PDUs which have been accepted and processed by the SNMP protocol entity.

::= { snmp 15 }

snmpInGetNexts OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of SNMP Get-Next PDUs which have been accepted and processed by the SNMP protocol entity.

::= { snmp 16 }

snmpInSetRequests OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of SNMP Set-Request PDUs which have been accepted and processed by the SNMP protocol entity.

::= { snmp 17 }

snmpInGetResponses OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of SNMP Get-Response PDUs which have been accepted and processed by the SNMP protocol entity.

::= { snmp 18 }

snmpInTraps OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of SNMP Trap PDUs which have been accepted and processed by the SNMP protocol entity.

::= { snmp 19 }

snmpOutTooBigs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of SNMP PDUs which were generated by the SNMP protocol entity and for which the value of the error-status field is tooBig.

::= { snmp 20 }

snmpOutNoSuchNames OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of SNMP PDUs which were generated by the SNMP protocol entity and for which the value of the error-status is noSuchName.

::= { snmp 21 }

snmpOutBadValues OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of SNMP PDUs which were generated by the SNMP protocol entity and for which the value of the error-status field is badValue.

::= { snmp 22 }

-- { snmp 23 } is not used

snmpOutGenErrs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of SNMP PDUs which were generated by the SNMP protocol entity and for which the value of the error-status field is genErr.

::= { snmp 24 }

snmpOutGetRequests OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of SNMP Get-Request PDUs which have been generated by the SNMP protocol entity.

::= { snmp 25 }

snmpOutGetNexts OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of SNMP Get-Next PDUs which have been generated by the SNMP protocol entity.

::= { snmp 26 }

snmpOutSetRequests OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of SNMP Set-Request PDUs which have been generated by the SNMP protocol entity.

::= { snmp 27 }

snmpOutGetResponses OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of SNMP Get-Response PDUs which have been generated by the SNMP protocol entity.

::= { snmp 28 }

snmpOutTraps OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

The total number of SNMP Trap PDUs which have been generated by the SNMP protocol entity.

::= { snmp 29 }

snmpEnableAuthenTraps OBJECT-TYPE

SYNTAX Integer { enabled(1), disabled(2) }

ACCESS read-write

STATUS mandatory

DESCRIPTION

Indicates whether the SNMP agent process is permitted to generate authentication-failure traps. The value of this object overrides any configuration information; as such, it provides a means whereby all authentication-failure traps may be disabled. Note that it is strongly recommended that this object be stored in non-volatile memory so that it remains constant between re-initializations of the network management system.

::= { snmp 30 }

HOST-RESOURCES-MIB

Note This is a reformatted version of HOST-RESOURCE-MIB. Download and compile all of the MIBs in this section from http://tools.cisco.com/Support/SNMP/do/BrowseMIB.do?local=en&step=2.

This MIB manages host systems. The term “host” means any computer that communicates with other similar computers attached to the internet and that is directly used by one or more human beings. Although this MIB does not necessarily apply to devices whose primary function is communications services (terminal servers, routers, bridges, and monitoring equipment), such relevance is not explicitly precluded. This MIB contains attributes that are common to all internet hosts including, for example, both personal computers and systems that run variants of Unix.

Before you can compile HOST-RESOURCES-MIB , you need to compile the MIBs listed below in the order listed.

1. SNMPv2-SMI
2. SNMPv2-TC
3. SNMPv2-CONF
4. SNMPv2-MIB
5. IANAifType-MIB
6. IF-MIB
7. RFC1155-SMI
8. RFC-1212
9. SNMPv2-SMI-v1
10. SNMPv2-TC-v1

Additional downloads are:

- OID File: HOST-RESOURCES-MIB.oid

HOST-RESOURCES-MIB Revisions

The following table lists the revisions to this MIB beginning with the latest revision.

Table 2: History of Revisions

Date	Action	Description
03-06-2000	Added and updated	Clarifications and bug fixes based on implementation experience. This revision was also reformatted in the SMIv2 format. The revisions made were: • Reformatted to new RFC document standards • Added copyright notice • Updated introduction to SNMP Framework • Updated references section • Added reference to RFC 2119 • Added a meaningful security considerations section

Date	Action	Description

Date	Action	Description
		New IANA considerations section for registration of new types, conversion to new SMIv2 syntax for the following types and macros: • Counter32, Integer32, Gauge32, MODULE-IDENTITY, OBJECT-TYPE, TEXTUAL-CONVENTION, OBJECT-IDENTITY, MODULE-COMPLIANCE, OBJECT-GROUP • Used new Textual Conventions: TruthValue, DateAndTime, AutonomousType, InterfaceIndexOrZero • Fixed typo in hrPrinterStatus • Added missing error bits to hrPrinterDetectedErrorState • Clarified confusion resulting from suggested mappings to hrPrinterStatus. • Clarified that size of objects of type InternationalDisplayString is number of octets, not number of encoded symbols. • Clarified the use of the following objects based on implementation experience: hrSystemInitialLoadDevice, hrSystemInitialLoadParameters, hrMemorySize, hrStorageSize, hrStorageAllocationFailures, hrDeviceErrors, hrProcessorLoad, hrNetworkIfIndex, hrDiskStorageCapacity, hrSWRunStatus, hrSWRunPerfCPU, and hrSWInstalledDate. • Clarified implementation technique for hrSWInstalledTable. • Used new AUGMENTS clause for hrSWRunPerfTable. • Added Internationalization

Date	Action	Description
		Considerations section. This revision published as RFC2790.
10-20-1999	Initial Version	The original version of this MIB, published as RFC1514. ::= { hrMIBAdminInfo 1 }

HOST-RESOURCES-MIB Definitions

The following definitions are imported for HOST-RESOURCES-MIB:

- MODULE-IDENTITY, OBJECT-TYPE, mib-2, Integer32, Counter32, Gauge32, TimeTicks
- From SNMPv2-SMI—TEXTUAL-CONVENTION, DisplayString, TruthValue, DateAndTime, AutonomousType
- From SNMPv2-TC—MODULE-COMPLIANCE, OBJECT-GROUP
- From SNMPv2-CONF—InterfaceIndexOrZero
- From IF-MIB—hostResourcesMibModule MODULE-IDENTITY

HOST-RESOURCES-MIB Object Identifiers

host OBJECT IDENTIFIER ::= { mib-2 25 }

hrSystem OBJECT IDENTIFIER ::= { host 1 }

hrStorage OBJECT IDENTIFIER ::= { host 2 }

hrDevice OBJECT IDENTIFIER ::= { host 3 }

hrSWRun OBJECT IDENTIFIER ::= { host 4 }

hrSWRunPerf OBJECT IDENTIFIER ::= { host 5 }

hrSWInstalled OBJECT IDENTIFIER ::= { host 6 }

hrMIBAdminInfo OBJECT IDENTIFIER ::= { host 7 }

Host Resources MIB Textual Conventions

KBytes ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

Storage size, expressed in units of 1024 bytes.

SYNTAX Integer32 (0..2147483647)

ProductID ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

This textual convention is intended to identify the manufacturer, model, and version of a specific hardware or software product. It is suggested that these OBJECT IDENTIFIERs are allocated such that all products from a particular manufacturer are registered under a subtree distinct to that manufacturer. In addition, all versions of a product should be registered under a subtree distinct to that product. With this strategy, a management station may uniquely determine the manufacturer and/or model of a product whose productID is unknown to the management station. Objects of this type may be useful for inventory purposes or for automatically detecting incompatibilities or version mismatches between various hardware and software components on a system.

For example, the product ID for the ACME 4860 66MHz clock doubled processor might be: enterprises.acme.acmeProcessors.a4860DX2.MHz66. A software product might be registered as: enterprises.acme.acmeOperatingSystems.acmeDOS.six(6).one(1).

SYNTAX OBJECT IDENTIFIER

UnknownProduct will be used for any unknown ProductID. UnknownProduct OBJECT IDENTIFIER ::= { 0 0 }.

InternationalDisplayString ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

This data type is used to model textual information in some character set. A network management station should use a local algorithm to determine which character set is in use and how it should be displayed. Note that this character set may be encoded with more than one octet per symbol, but will most often be NVT ASCII. When a size clause is specified for an object of this type, the size refers to the length in octets, not the number of symbols.

SYNTAX OCTET STRING

Host Resources System Group

hrSystemUptime OBJECT-TYPE

SYNTAX TimeTicks

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The amount of time since this host was last initialized. Note that this is different from sysUpTime in the SNMPv2-MIB [RFC1907] because sysUpTime is the uptime of the network management portion of the system.

::= { hrSystem 1 }

hrSystemDate OBJECT-TYPE

SYNTAX DateAndTime

MAX-ACCESS read-write

STATUS current

DESCRIPTION

The host's notion of the local date and time of day.

::= { hrSystem 2 }

hrSystemInitialLoadDevice OBJECT-TYPE

SYNTAX Integer32 (1..2147483647)

MAX-ACCESS read-write

STATUS current

DESCRIPTION

The index of the hrDeviceEntry for the device from which this host is configured to load its initial operating system configuration (i.e., which operating system code and/or boot parameters). Note that writing to this object just changes the configuration that will be used the next time the operating system is loaded and does not actually cause the reload to occur.

::= { hrSystem 3 }

hrSystemInitialLoadParameters OBJECT-TYPE

SYNTAX InternationalDisplayString (SIZE (0..128))

MAX-ACCESS read-write

STATUS current

DESCRIPTION

This object contains the parameters (e.g. a pathname and parameter) supplied to the load device when requesting the initial operating system configuration from that device. Note that writing to this object just changes the configuration that will be used the next time the operating system is loaded and does not actually cause the reload to occur.

::= { hrSystem 4 }

hrSystemNumUsers OBJECT-TYPE

SYNTAX Gauge32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The number of user sessions for which this host is storing state information. A session is a collection

of processes requiring a single act of user authentication and possibly subject to collective job control.

::= { hrSystem 5 }

hrSystemProcesses OBJECT-TYPE

SYNTAX Gauge32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The number of process contexts currently loaded or running on this system.

::= { hrSystem 6 }

hrSystemMaxProcesses OBJECT-TYPE

SYNTAX Integer32 (0..2147483647)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The maximum number of process contexts this system can support. If there is no fixed maximum, the value should be zero. On systems that have a fixed maximum, this object can help diagnose failures that occur when this maximum is reached.

::= { hrSystem 7 }

Host Resources Storage Group

Registration point for storage types, for use with hrStorageType. These are defined in the HOST-RESOURCES-TYPES module.

hrStorageTypes OBJECT IDENTIFIER ::= { hrStorage 1 }

hrMemorySize OBJECT-TYPE

SYNTAX KBytes

UNITS KBytes

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The amount of physical read-write main memory, typically RAM, contained by the host.

::= { hrStorage 2 }

hrStorageTable OBJECT-TYPE

SYNTAX Sequence of HrStorageEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

The (conceptual) table of logical storage areas on the host. An entry shall be placed in the storage table for each logical area of storage that is allocated and has fixed resource limits. The amount of storage represented in an entity is the amount actually usable by the requesting entity, and excludes loss due to formatting or file system reference information.

These entries are associated with logical storage areas, as might be seen by an application, rather than physical storage entities which are typically seen by an operating system. Storage such as tapes and floppies without file systems on them are typically not allocated in chunks by the operating system to

requesting applications, and therefore shouldn't appear in this table. Examples of valid storage for this table include disk partitions, file systems, RAM (for some architectures this is further segmented into regular memory, extended memory, and so on), backing store for virtual memory (`swap space').

This table is intended to be a useful diagnostic for “out of memory” and “out of buffers” types of failures. In addition, it can be a useful performance monitoring tool for tracking memory, disk, or buffer usage.

::= { hrStorage 3 }

hrStorageEntry OBJECT-TYPE

SYNTAX HrStorageEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

A (conceptual) entry for one logical storage area on the host. As an example, an instance of the hrStorageType object might be named hrStorageType.3

INDEX { hrStorageIndex }

::= { hrStorageTable 1 }

hrStorageEntry ::= SEQUENCE { hrStorageIndex Integer32, hrStorageTypeAutonomousType, hrStorageDescr DisplayString, hrStorageAllocationUnits Integer32, hrStorageSizeInteger32, hrStorageUsedInteger32, hrStorageAllocationFailures Counter32 }

hrStorageIndex OBJECT-TYPE

SYNTAX Integer32 (1..2147483647)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

A unique value for each logical storage area contained by the host.

::= { hrStorageEntry 1 }

hrStorageType OBJECT-TYPE

SYNTAX AutonomousType

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The type of storage represented by this entry.

::= { hrStorageEntry 2 }

hrStorageDescr OBJECT-TYPE

SYNTAX DisplayString

MAX-ACCESS read-only

STATUS current

DESCRIPTION

A description of the type and instance of the storage described by this entry.

::= { hrStorageEntry 3 }

hrStorageAllocationUnits OBJECT-TYPE

SYNTAX Integer32 (1..2147483647)

UNITS Bytes

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The size, in bytes, of the data objects allocated from this pool. If this entry is monitoring sectors, blocks, buffers, or packets, for example, this number will commonly be greater than one. Otherwise this number will typically be one.

::= { hrStorageEntry 4 }

hrStorageSize OBJECT-TYPE

SYNTAX Integer32 (0..2147483647)

MAX-ACCESS read-write

STATUS current

DESCRIPTION

The size of the storage represented by this entry, in units of hrStorageAllocationUnits. This object is writable to allow remote configuration of the size of the storage area in those cases where such an operation makes sense and is possible on the underlying system. For example, the amount of main memory allocated to a buffer pool might be modified or the amount of disk space allocated to virtual memory might be modified.

::= { hrStorageEntry 5 }

hrStorageUsed OBJECT-TYPE

SYNTAX Integer32 (0..2147483647)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The amount of the storage represented by this entry that is allocated, in units of hrStorageAllocationUnits.

::= { hrStorageEntry 6 }

hrStorageAllocationFailures OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The number of requests for storage represented by this entry that could not be honored due to not enough storage. It should be noted that as this object has a SYNTAX of Counter32, that it does not have a defined initial value. However, it is recommended that this object be initialized to zero, even though management stations must not depend on such an initialization.

::= { hrStorageEntry 7 }

Host Resources Device Group

The device group is useful for identifying and diagnosing the devices on a system. The hrDeviceTable contains common information for any type of device. In addition, some devices have device-specific tables for more detailed information. More such tables may be defined in the future for other device types. Registration point for device types, for use with hrDeviceType. These are defined in the HOST-RESOURCES-TYPES module.

hrDeviceTypes OBJECT IDENTIFIER ::= { hrDevice 1 }

hrDeviceTable OBJECT-TYPE

SYNTAX Sequence of hrDeviceEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

The (conceptual) table of devices contained by the host.

::= { hrDevice 2 }

hrDeviceEntry OBJECT-TYPE

SYNTAX hrDeviceEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

A (conceptual) entry for one device contained by the host. As an example, an instance of the hrDeviceType object might be named hrDeviceType.3

INDEX { hrDeviceIndex }

::= { hrDeviceTable 1 }

HrDeviceEntry ::= SEQUENCE { hrDeviceIndex Integer32, hrDeviceTypeAutonomousType, hrDeviceDescr DisplayString, hrDeviceID ProductID, hrDeviceStatus INTEGER, hrDeviceErrors Counter32 }

hrDeviceIndex OBJECT-TYPE

SYNTAX Integer32 (1..2147483647)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

A unique value for each device contained by the host. The value for each device must remain constant at least from one re-initialization of the agent to the next re-initialization.

::= { hrDeviceEntry 1 }

hrDeviceType OBJECT-TYPE

SYNTAX AutonomousType

MAX-ACCESS read-only

STATUS current

DESCRIPTION

An indication of the type of device. If this value is “hrDeviceProcessor { hrDeviceTypes 3 }” then an entry exists in the hrProcessorTable which corresponds to this device. If this value is “hrDeviceNetwork { hrDeviceTypes 4 }”, then an entry exists in the hrNetworkTable which corresponds to this device. If this value is “hrDevicePrinter { hrDeviceTypes 5 }”, then an entry exists in the hrPrinterTable which corresponds to this device.

If this value is “hrDeviceDiskStorage { hrDeviceTypes 6 }”, then an entry exists in the hrDiskStorageTable which corresponds to this device.

::= { hrDeviceEntry 2 }

hrDeviceDescr OBJECT-TYPE

SYNTAX DisplayString (SIZE (0..64))

MAX-ACCESS read-only

STATUS current

DESCRIPTION

A textual description of this device, including the device's manufacturer and revision, and optionally, its serial number.

::= { hrDeviceEntry 3 }

hrDeviceID OBJECT-TYPE

SYNTAX ProductID

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The product ID for this device.

::= { hrDeviceEntry 4 }

hrDeviceStatus OBJECT-TYPE

SYNTAX INTEGER { unknown(1), running(2), warning(3), testing(4), down(5) }

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The current operational state of the device described by this row of the table. A value unknown(1) indicates that the current state of the device is unknown. running(2) indicates that the device is up and running and that no unusual error conditions are known. The warning(3) state indicates that agent has

been informed of an unusual error condition by the operational software (e.g., a disk device driver) but that the device is still 'operational'. An example would be a high number of soft errors on a disk. A value of testing(4), indicates that the device is not available for use because it is in the testing state. The state of down(5) is used only when the agent has been informed that the device is not available for any use.

::= { hrDeviceEntry 5 }

hrDeviceErrors OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The number of errors detected on this device. It should be noted that as this object has a SYNTAX of Counter32, that it does not have a defined initial value. However, it is recommended that this object be initialized to zero, even though management stations must not depend on such an initialization.

::= { hrDeviceEntry 6 }

hrProcessorTable OBJECT-TYPE

SYNTAX Sequence of hrProcessorEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

The (conceptual) table of processors contained by the host. Note that this table is potentially sparse: a (conceptual) entry exists only if the correspondent value of the hrDeviceType object is hrDeviceProcessor.

::= { hrDevice 3 }

hrProcessorEntry OBJECT-TYPE

SYNTAX hrProcessorEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

A (conceptual) entry for one processor contained by the host. The hrDeviceIndex in the index represents the entry in the hrDeviceTable that corresponds to the hrProcessorEntry. As an example of how objects in this table are named, an instance of the hrProcessorFrwID object might be named hrProcessorFrwID.3

INDEX { hrDeviceIndex }

::= { hrProcessorTable 1 }

HrProcessorEntry ::= SEQUENCE { hrProcessorFrwIDProductID, hrProcessorLoad Integer32 }

hrProcessorFrwID OBJECT-TYPE

SYNTAX ProductID

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The product ID of the firmware associated with the processor.

::= { hrProcessorEntry 1 }

hrProcessorLoad OBJECT-TYPE

SYNTAX Integer32 (0..100)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The average, over the last minute, of the percentage of time that this processor was not idle. Implementations may approximate this one minute smoothing period if necessary.

::= { hrProcessorEntry 2 }

hrNetworkTable OBJECT-TYPE

SYNTAX Sequence of hrNetworkEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

The (conceptual) table of network devices contained by the host. Note that this table is potentially sparse: a (conceptual) entry exists only if the correspondent value of the hrDeviceType object is hrDeviceNetwork.

::= { hrDevice 4 }

hrNetworkEntry OBJECT-TYPE

SYNTAX hrNetworkEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

A (conceptual) entry for one network device contained by the host. The hrDeviceIndex in the index represents the entry in the hrDeviceTable that corresponds to the hrNetworkEntry. As an example of how objects in this table are named, an instance of the hrNetworkIfIndex object might be named hrNetworkIfIndex.3.

INDEX { hrDeviceIndex }

::= { hrNetworkTable 1 }

hrNetworkEntry ::= SEQUENCE { hrNetworkIfIndexInterfaceIndexOrZero }

hrNetworkIfIndex OBJECT-TYPE

SYNTAX InterfaceIndexOrZero

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The value of ifIndex which corresponds to this network device. If this device is not represented in the ifTable, then this value shall be zero.

::= { hrNetworkEntry 1 }

hrPrinterTable OBJECT-TYPE

SYNTAX Sequence of hrPrinterEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

The (conceptual) table of printers local to the host. Note that this table is potentially sparse: a (conceptual) entry exists only if the correspondent value of the hrDeviceType object is hrDevicePrinter.

::= { hrDevice 5 }

hrPrinterEntry OBJECT-TYPE

SYNTAX hrPrinterEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

A (conceptual) entry for one printer local to the host. The hrDeviceIndex in the index represents the entry in the hrDeviceTable that corresponds to the hrPrinterEntry.

As an example of how objects in this table are named, an instance of the hrPrinterStatus object might be named hrPrinterStatus.3

INDEX { hrDeviceIndex }

::= { hrPrinterTable 1 }

hrPrinterEntry ::= SEQUENCE { hrPrinterStatus INTEGER, hrPrinterDetectedErrorState OCTET STRING }

hrPrinterStatus OBJECT-TYPE

SYNTAX INTEGER { other(1), unknown(2), idle(3), printing(4), warmup(5) }

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The current status of this printer device.

::= { hrPrinterEntry 1 }

hrPrinterDetectedErrorState OBJECT-TYPE

SYNTAX OCTET STRING

MAX-ACCESS read-only

STATUS current

DESCRIPTION

This object represents any error conditions detected by the printer. The error conditions are encoded as bits in an octet string, with the following definitions (condition first then bit number):

- lowPaper 0
- noPaper 1
- lowToner 2
- noToner 3
- doorOpen 4
- jammed5
- offline 6
- serviceRequested 7
- inputTrayMissing 8
- outputTrayMissing 9
- markerSupplyMissing 10
- outputNearFull 11
- outputFull 12
- inputTrayEmpty 13
- overduePreventMaint 14

Bits are numbered starting with the most significant bit of the first byte being bit 0, the least significant bit of the first byte being bit 7, the most significant bit of the second byte being bit 8, and so on. A one bit encodes that the condition was detected, while a zero bit encodes that the condition was not detected.

This object is useful for alerting an operator to specific warning or error conditions that may occur, especially those requiring human intervention.

::= { hrPrinterEntry 2 }

hrDiskStorageTable OBJECT-TYPE

SYNTAX Sequence of hrDiskStorageEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

The (conceptual) table of long-term storage devices contained by the host. In particular, disk devices accessed remotely over a network are not included here. Note that this table is potentially sparse: a (conceptual) entry exists only if the correspondent value of the hrDeviceType object is hrDeviceDiskStorage.

::= { hrDevice 6 }

hrDiskStorageEntry OBJECT-TYPE

SYNTAX hrDiskStorageEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

A (conceptual) entry for one long-term storage device contained by the host. The hrDeviceIndex in the index represents the entry in the hrDeviceTable that corresponds to the hrDiskStorageEntry. As an example, an instance of the hrDiskStorageCapacity object might be named hrDiskStorageCapacity.3

INDEX { hrDeviceIndex }

::= { hrDiskStorageTable 1 }

hrDiskStorageEntry ::= SEQUENCE { hrDiskStorageAccess INTEGER, hrDiskStorageMedia INTEGER, hrDiskStorageRemoveble TruthValue, hrDiskStorageCapacity KBytes }

hrDiskStorageAccess OBJECT-TYPE

SYNTAX INTEGER { readWrite(1), readOnly(2) }

MAX-ACCESS read-only

STATUS current

DESCRIPTION

An indication if this long-term storage device is readable and writable or only readable. This should reflect the media type, any write-protect mechanism, and any device configuration that affects the entire device.

::= { hrDiskStorageEntry 1 }

hrDiskStorageMedia OBJECT-TYPE

SYNTAX INTEGER { other(1), unknown(2), hardDisk(3), floppyDisk(4), opticalDiskROM(5), opticalDiskWORM(6), --Write Once Read Many-- opticalDiskRW(7), ramDisk(8) }

MAX-ACCESS read-only

STATUS current

DESCRIPTION

An indication of the type of media used in this long-term storage device.

::= { hrDiskStorageEntry 2 }

hrDiskStorageRemoveble OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-only

STATUS current

DESCRIPTION

Denotes whether or not the disk media may be removed from the drive.

::= { hrDiskStorageEntry 3 }

hrDiskStorageCapacity OBJECT-TYPE

SYNTAX KBytes

UNITS KBytes

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The total size for this long-term storage device. If the media is removable and is currently removed, this value should be zero.

::= { hrDiskStorageEntry 4 }

hrPartitionTable OBJECT-TYPE

SYNTAX Sequence of hrPartitionEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

The (conceptual) table of partitions for long-term storage devices contained by the host. In particular, partitions accessed remotely over a network are not included here.

::= { hrDevice 7 }

hrPartitionEntry OBJECT-TYPE

SYNTAX hrPartitionEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

A (conceptual) entry for one partition. The hrDeviceIndex in the index represents the entry in the hrDeviceTable that corresponds to the hrPartitionEntry.

As an example of how objects in this table are named, an instance of the hrPartitionSize object might be named hrPartitionSize.3.1

INDEX { hrDeviceIndex, hrPartitionIndex }

::= { hrPartitionTable 1 }

hrPartitionEntry ::= SEQUENCE { hrPartitionIndexInteger32, hrPartitionLabelInternationalDisplayString, hrPartitionID OCTET STRING, hrPartitionSize Bytes, hrPartitionFSIndex Integer32 }

hrPartitionIndex OBJECT-TYPE

SYNTAX Integer32 (1..2147483647)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

A unique value for each partition on this long-term storage device. The value for each long-term storage device must remain constant at least from one re-initialization of the agent to the next re-initialization.

::= { hrPartitionEntry 1 }

hrPartitionLabel OBJECT-TYPE

SYNTAX InternationalDisplayString (SIZE (0..128))

MAX-ACCESS read-only

STATUS current

DESCRIPTION

A textual description of this partition.

::= { hrPartitionEntry 2 }

hrPartitionID OBJECT-TYPE

SYNTAX OCTET STRING

MAX-ACCESS read-only

STATUS current

DESCRIPTION

A descriptor which uniquely represents this partition to the responsible operating system. On some systems, this might take on a binary representation.

::= { hrPartitionEntry 3 }

hrPartitionSize OBJECT-TYPE

SYNTAX KBytes

UNITS KBytes

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The size of this partition.

::= { hrPartitionEntry 4 }

hrPartitionFSIndex OBJECT-TYPE

SYNTAX Integer32 (0..2147483647)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The index of the file system mounted on this partition. If no file system is mounted on this partition, then this value shall be zero. Note that multiple partitions may point to one file system, denoting that that file system resides on those partitions. Multiple file systems may not reside on one partition.

::= { hrPartitionEntry 5 }

File System Table

Registration point for popular File System types, for use with hrFSType. These are defined in the HOST-RESOURCES-TYPES module.

hrFSTypes OBJECT IDENTIFIER

::= { hrDevice 9 }

hrFSTable OBJECT-TYPE

SYNTAX Sequence of hrFSEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

The (conceptual) table of file systems local to this host or remotely mounted from a file server. File systems that are in only one user's environment on a multi-user system will not be included in this table.

::= { hrDevice 8 }

hrFSEntry OBJECT-TYPE

SYNTAX hrFSEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

A (conceptual) entry for one file system local to this host or remotely mounted from a file server. File systems that are in only one user's environment on a multi-user system will not be included in this table.

As an example of how objects in this table are named, an instance of the hrFSMountPoint object might be named hrFSMountPoint.3

INDEX { hrFSIndex }

::= { hrFSTable 1 }

hrFSEntry ::= SEQUENCE { hrFSIndex Integer32, hrFSMountPoint InternationalDisplayString, hrFSRemoteMountPointInternationalDisplayString, hrFSTypeAutonomousType, hrFSAccess INTEGER, hrFSBootableTruthValue, hrFSStorageIndexInteger32, hrFSLastFullBackupDate DateAndTime, hrFSLastPartialBackupDate DateAndTime }

hrFSIndex OBJECT-TYPE

SYNTAX Integer32 (1..2147483647)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

A unique value for each file system local to this host. The value for each file system must remain constant at least from one re-initialization of the agent to the next re-initialization.

::= { hrFSEntry 1 }

hrFSMountPoint OBJECT-TYPE

SYNTAX InternationalDisplayString (SIZE(0..128))

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The path name of the root of this file system.

::= { hrFSEntry 2 }

hrFSRemoteMountPoint OBJECT-TYPE

SYNTAX InternationalDisplayString (SIZE(0..128))

MAX-ACCESS read-only

STATUS current

DESCRIPTION

A description of the name and/or address of the server that this file system is mounted from. This may also include parameters such as the mount point on the remote file system. If this is not a remote file system, this string should have a length of zero.

::= { hrFSEntry 3 }

hrFSType OBJECT-TYPE

SYNTAX AutonomousType

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The value of this object identifies the type of this file system.

::= { hrFSEntry 4 }

hrFSAccess OBJECT-TYPE

SYNTAX Integer { readWrite(1), readOnly(2) }

MAX-ACCESS read-only

STATUS current

DESCRIPTION

An indication if this file system is logically configured by the operating system to be readable and writable or only readable. This does not represent any local access-control policy, except one that is applied to the file system as a whole.

::= { hrFSEntry 5 }

hrFSBootable OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-only

STATUS current

DESCRIPTION

A flag indicating whether this file system is bootable.

::= { hrFSEntry 6 }

hrFSStorageIndex OBJECT-TYPE

SYNTAX Integer32 (0..2147483647)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The index of the hrStorageEntry that represents information about this file system. If there is no such information available, then this value shall be zero. The relevant storage entry will be useful in tracking the percent usage of this file system and diagnosing errors that may occur when it runs out of space.

::= { hrFSEntry 7 }

hrFSLastFullBackupDate OBJECT-TYPE

SYNTAX DateAndTime

MAX-ACCESS read-write

STATUS current

DESCRIPTION

The last date at which this complete file system was copied to another storage device for backup. This information is useful for ensuring that backups are being performed regularly. If this information is not known, then this variable shall have the value corresponding to January 1, year 0000, 00:00:00.0, which is encoded as (hex) 00 00 01 01 00 00 00 00.

::= { hrFSEntry 8 }

hrFSLastPartialBackupDate OBJECT-TYPE

SYNTAX DateAndTime

MAX-ACCESS read-write

STATUS current

DESCRIPTION

The last date at which a portion of this file system was copied to another storage device for backup. This information is useful for ensuring that backups are being performed regularly. If this information is not known, then this variable shall have the value corresponding to January 1, year 0000, 00:00:00.0, which is encoded as (hex) 00 00 01 01 00 00 00 00.

::= { hrFSEntry 9 }

Host Resources Running Software Group

The hrSWRunTable contains an entry for each distinct piece of software that is running or loaded into physical or virtual memory in preparation for running. This includes the host's operating system, device drivers, and applications.

hrSWOSIndex OBJECT-TYPE

SYNTAX Integer32 (1..2147483647)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The value of the hrSWRunIndex for the hrSWRunEntry that represents the primary operating system running on this host. This object is useful for quickly and uniquely identifying that primary operating system.

::= { hrSWRun 1 }

hrSWRunTable OBJECT-TYPE

SYNTAX Sequence of hrSWRunEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

The (conceptual) table of software running on the host.

::= { hrSWRun 2 }

hrSWRunEntry OBJECT-TYPE

SYNTAX hrSWRunEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

A (conceptual) entry for one piece of software running on the host Note that because the installed software table only contains information for software stored locally on this host, not every piece of running software will be found in the installed software table. This is true of software that was loaded and run from a non-local source, such as a network-mounted file system.

As an example of how objects in this table are named, an instance of the hrSWRunName object might be named hrSWRunName.1287

INDEX { hrSWRunIndex }

::= { hrSWRunTable 1 }

HrSWRunEntry ::= SEQUENCE { hrSWRunIndex Integer32, hrSWRunNameInternationalDisplayString, hrSWRunID ProductID, hrSWRunPathInternationalDisplayString, hrSWRunParameters InternationalDisplayString, hrSWRunTypeINTEGER, hrSWRunStatus INTEGER }

hrSWRunIndex OBJECT-TYPE

SYNTAX Integer32 (1..2147483647)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

A unique value for each piece of software running on the host. Wherever possible, this should be the system's native, unique identification number.

::= { hrSWRunEntry 1 }

hrSWRunName OBJECT-TYPE

SYNTAX InternationalDisplayString (SIZE (0..64))

MAX-ACCESS read-only

STATUS current

DESCRIPTION

A textual description of this running piece of software, including the manufacturer, revision, and the name by which it is commonly known. If this software was installed locally, this should be the same string as used in the corresponding hrSWInstalledName.

::= { hrSWRunEntry 2 }

hrSWRunID OBJECT-TYPE

SYNTAX ProductID

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The product ID of this running piece of software.

::= { hrSWRunEntry 3 }

hrSWRunPath OBJECT-TYPE

SYNTAX InternationalDisplayString (SIZE(0..128))

MAX-ACCESS read-only

STATUS current

DESCRIPTION

A description of the location on long-term storage (e.g. a disk drive) from which this software was loaded.

::= { hrSWRunEntry 4 }

hrSWRunParameters OBJECT-TYPE

SYNTAX InternationalDisplayString (SIZE(0..128))

MAX-ACCESS read-only

STATUS current

DESCRIPTION

A description of the parameters supplied to this software when it was initially loaded.

::= { hrSWRunEntry 5 }

hrSWRunType OBJECT-TYPE

SYNTAX INTEGER { unknown(1), operatingSystem(2), deviceDriver(3), application(4) }

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The type of this software.

::= { hrSWRunEntry 6 }

hrSWRunStatus OBJECT-TYPE

SYNTAX INTEGER { running(1), runnable(2),-- waiting for resource -- (i.e., CPU, memory, IO) notRunnable(3), -- loaded but waiting for event invalid(4) -- not loaded }

MAX-ACCESS read-write

STATUS current

DESCRIPTION

The status of this running piece of software. Setting this value to invalid(4) shall cause this oftware to stop running and to be unloaded. Sets to other values are not valid.

::= { hrSWRunEntry 7 }

Host Resources Running Software Performance Group

The hrSWRunPerfTable contains an entry corresponding to each entry in the hrSWRunTable.

hrSWRunPerfTable OBJECT-TYPE

SYNTAX Sequence of hrSWRunPerfEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

The (conceptual) table of running software performance metrics.

::= { hrSWRunPerf 1 }

hrSWRunPerfEntry OBJECT-TYPE

SYNTAX hrSWRunPerfEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

A (conceptual) entry containing software performance metrics. As an example, an instance of the hrSWRunPerfCPU object might be named hrSWRunPerfCPU.1287. This table augments information in the hrSWRunTable.

AUGMENTS { hrSWRunEntry }

::= { hrSWRunPerfTable 1 }

hrSWRunPerfEntry ::= SEQUENCE { hrSWRunPerfCPU Integer32, hrSWRunPerfMem KBytes }

hrSWRunPerfCPU OBJECT-TYPE

SYNTAX Integer32 (0..2147483647)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The number of centi-seconds of the total system's CPU resources consumed by this process. Note that on a multi-processor system, this value may increment by more than one centi-second in one centi-second of real (wall clock) time.

::= { hrSWRunPerfEntry 1 }

hrSWRunPerfMem OBJECT-TYPE

SYNTAX KBytes

UNITS KBytes

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The total amount of real system memory allocated to this process.

::= { hrSWRunPerfEntry 2 }

Host Resources Installed Software Group

The hrSWInstalledTable contains an entry for each piece of software installed in long-term storage (e.g. a disk drive) locally on this host. Note that this does not include software loadable remotely from a network server. Different implementations may track software in varying ways. For example, while some implementations may track executable files as distinct pieces of software, other implementations may use other strategies such as keeping track of software packages (e.g., related groups of files) or keeping track of system or application patches.

This table is useful for identifying and inventoring software on a host and for diagnosing incompatibility and version mismatch problems between various pieces of hardware and software.

hrSWInstalledLastChange OBJECT-TYPE

SYNTAX TimeTicks

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The value of sysUpTime when an entry in the hrSWInstalledTable was last added, renamed, or deleted. Because this table is likely to contain many entries, polling of this object allows a management station to determine when re-downloading of the table might be useful.

::= { hrSWInstalled 1 }

hrSWInstalledLastUpdateTime OBJECT-TYPE

SYNTAX TimeTicks

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The value of sysUpTime when the hrSWInstalledTable was last completely updated. Because caching of this data will be a popular implementation strategy, retrieval of this object allows a management station to obtain a guarantee that no data in this table is older than the indicated time.

::= { hrSWInstalled 2 }

hrSWInstalledTable OBJECT-TYPE

SYNTAX SEQUENCE OF HrSWInstalledEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

The (conceptual) table of software installed on this host.

::= { hrSWInstalled 3 }

hrSWInstalledEntry OBJECT-TYPE

SYNTAX HrSWInstalledEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

A (conceptual) entry for a piece of software installed on this host. As an example of how objects in this table are named, an instance of the hrSWInstalledName object might be named hrSWInstalledName.96

INDEX { hrSWInstalledIndex }

::= { hrSWInstalledTable 1 }

hrSWInstalledEntry ::= SEQUENCE { hrSWInstalledIndex Integer32,
hrSWInstalledNameInternationalDisplayString, hrSWInstalledID ProductID,
hrSWInstalledTypeINTEGER, hrSWInstalledDateDateAndTime }

hrSWInstalledIndex OBJECT-TYPE

SYNTAX Integer32 (1..2147483647)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

A unique value for each piece of software installed on the host. This value shall be in the range from 1 to the number of pieces of software installed on the host.

::= { hrSWInstalledEntry 1 }

hrSWInstalledName OBJECT-TYPE

SYNTAX InternationalDisplayString (SIZE (0..64))

MAX-ACCESS read-only

STATUS current

DESCRIPTION

A textual description of this installed piece of software, including the manufacturer, revision, the name by which it is commonly known, and optionally, its serial number.

::= { hrSWInstalledEntry 2 }

hrSWInstalledID OBJECT-TYPE

SYNTAX ProductID

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The product ID of this installed piece of software.

::= { hrSWInstalledEntry 3 }

hrSWInstalledType OBJECT-TYPE

SYNTAX INTEGER { unknown(1), operatingSystem(2), deviceDriver(3), application(4) }

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The type of this software.

::= { hrSWInstalledEntry 4 }

hrSWInstalledDate OBJECT-TYPE

SYNTAX DateAndTime

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The last-modification date of this application as it would appear in a directory listing.

If this information is not known, then this variable shall have the value corresponding to January 1, year 0000, 00:00:00.0, which is encoded as (hex) 00 00 01 01 00 00 00 00.

::= { hrSWInstalledEntry 5 }

Host Resources Conformance Information

hrMIBCompliances OBJECT IDENTIFIER

::= { hrMIBAdminInfo 2 }

hrMIBGroups OBJECT IDENTIFIER

::= { hrMIBAdminInfo 3 }

Host Resources Compliance Statements

hrMIBCompliance MODULE-COMPLIANCE

STATUS current

DESCRIPTION

The requirements for conformance to the Host Resources MIB.

MANDATORY-GROUPS { hrSystemGroup, hrStorageGroup, hrDeviceGroup }

OBJECT hrSystemDate

MIN-ACCESS read-only

DESCRIPTION

Write access is not required.

OBJECT hrSystemInitialLoadDevice

MIN-ACCESS read-only

DESCRIPTION

Write access is not required.

OBJECT hrSystemInitialLoadParameters

MIN-ACCESS read-only

DESCRIPTION

Write access is not required.

OBJECT hrStorageSize

MIN-ACCESS read-only

DESCRIPTION

Write access is not required.

OBJECT hrFSLastFullBackupDate

MIN-ACCESS read-only

DESCRIPTION Write access is not required.

OBJECT hrFSLastPartialBackupDate

MIN-ACCESS read-only

DESCRIPTION

Write access is not required.

GROUP hrSWRunGroup

DESCRIPTION

The Running Software Group. Implementation of this group is mandatory only when the hrSWRunPerfGroup is implemented.

OBJECT hrSWRunStatus

MIN-ACCESS read-only

DESCRIPTION

Write access is not required.

GROUP hrSWRunPerfGroup

DESCRIPTION

The Running Software Performance Group. Implementation of this group is at the discretion of the implementor.

GROUP hrSWInstalledGroup

DESCRIPTION

The Installed Software Group. Implementation of this group is at the discretion of the implementor.

::= { hrMIBCompliances 1 }

hrSystemGroup OBJECT-GROUP

OBJECTS { hrSystemUptime, hrSystemDate, hrSystemInitialLoadDevice, hrSystemInitialLoadParameters, hrSystemNumUsers, hrSystemProcesses, hrSystemMaxProcesses }

STATUS current

DESCRIPTION

he Host Resources System Group.

::= { hrMIBGroups 1 }

hrStorageGroup OBJECT-GROUP

OBJECTS { hrMemorySize, hrStorageIndex, hrStorageType, hrStorageDescr, hrStorageAllocationUnits, hrStorageSize, hrStorageUsed, hrStorageAllocationFailures }

STATUS current

DESCRIPTION

The Host Resources Storage Group.

::= { hrMIBGroups 2 }

hrDeviceGroup OBJECT-GROUP

OBJECTS { hrDeviceIndex, hrDeviceType, hrDeviceDescr, hrDeviceID, hrDeviceStatus, hrDeviceErrors, hrProcessorFrwID, hrProcessorLoad, hrNetworkIfIndex, hrPrinterStatus, hrPrinterDetectedErrorState, hrDiskStorageAccess, hrDiskStorageMedia, hrDiskStorageRemoveble, hrDiskStorageCapacity, hrPartitionIndex, hrPartitionLabel, hrPartitionID, hrPartitionSize, hrPartitionFSIndex, hrFSIndex, hrFSMountPoint, hrFSRemoteMountPoint, hrFSType, hrFSAccess, hrFSBootable, hrFSStorageIndex, hrFSLastFullBackupDate, hrFSLastPartialBackupDate }

STATUS current

DESCRIPTION

The Host Resources Device Group.

::= { hrMIBGroups 3 }

hrSWRunGroup OBJECT-GROUP

OBJECTS {hrSWOSIndex, hrSWRunIndex, hrSWRunName, hrSWRunID, hrSWRunPath, hrSWRunParameters, hrSWRunType, hrSWRunStatus }

STATUS current

DESCRIPTION

The Host Resources Running Software Group.

::= { hrMIBGroups 4 }

hrSWRunPerfGroup OBJECT-GROUP

OBJECTS { hrSWRunPerfCPU, hrSWRunPerfMem }

STATUS current

DESCRIPTION

The Host Resources Running Software Performance Group.

::= { hrMIBGroups 5 }

hrSWInstalledGroup OBJECT-GROUP

OBJECTS { hrSWInstalledLastChange, hrSWInstalledLastUpdateTime, hrSWInstalledIndex, hrSWInstalledName, hrSWInstalledID, hrSWInstalledType, hrSWInstalledDate }

STATUS current

DESCRIPTION

The Host Resources Installed Software Group.

::= { hrMIBGroups 6 }

Cisco Unified CM Release 6.x Feature Services

The following table lists the Cisco Unified Serviceability feature services in Unified Communications Manager Release 6.x. It also lists the applicable HOST-RESOURCES-MIB OIDs, clearing values, and object responses.

Table 3: Unified Communications Manager Release 6.x Feature Services and HOST-RESOURCES-MIB

Cisco Unified CM Release 6.x Feature Services	hrSWRunNameOIDs	Clearing Values(Positive String)	Object Responses
Cisco Unified CM Attendant Console Server Service	1.3.6.1.2.1.25.4.2.1.2	acserver	Cisco CallManager Attendant Console Server Service Failure
Cisco Extended Functions Service		cef	Cisco Extended Functions Service Failure
Cisco Serviceability Reporter service		rtmtreporter	Cisco Serviceability Reporter service failure
Compaq Insite Manager Service		cmascsid	Compaq Insite Manager Service Failure

Cisco Unified CM Release 6.x Feature Services	hrSWRunNameOIDs	Clearing Values(Positive String)	Object Responses
Cisco Messaging Interface Service		cmi	Cisco Messaging Interface Service Failure
CSA service		ciscosecd	Cisco Security Agent Service Failure
CISCO-CCM-MIB activation on system	1.3.6.1.4.1.9.9.156	ccmAgt	CCM MIB Query Capabilities Disabled
IP Voice Media Streaming Service IF ACTIVATED	1.3.6.1.2.1.25.4.2.1.2	ipvmsd	IP Voice Media Streaming Service Failure
Cisco Unified CM Service If Activated		ccm	Cisco CallManager Service Failure
TFTP Service If Activated		ctftp	TFTP Service Failure
CTIManager Service If Activated		CTIManager	CTIManager Service Failure
Syslog Service		syslogd	Syslog Service Failure
DHCP Monitor Service If Activated		DHCP Monitor	DHCPMonitor Service Failure
Certificate Trust List Service Availability If Activated		CTLProvider	CTLProvider Service Failure
Certificate Authority Proxy Function Service Availability If Activated		capf	Certificate Authority Proxy Function Failure
DirSync Service Availability If Activated		CCMDirSync	CCMDirSync Service Failure
HOST-RESOURCES MIB activation on system	1.3.6.1.2.1.25	host_agent.pl	Host MIB Query Capabilities Disabled
MIB2 (RFC1213) activation on system	1.3.6.1.2.1	mib2_agent.pl	MIB2 MIB Query Capabilities Disabled
SYSAPPL-MIB activation on system	1.3.6.1.2.1.54	sapp_agent.pl	SysApp MIB Query Capabilities Disabled

Cisco Unified CM Release 6.x Network Services

The following table lists the Cisco Unified Serviceability network services in Unified Communications Manager Release 6.x. It also lists the applicable HOST-RESOURCES-MIB OIDs, clearing values, and object responses.

Table 4: Unified Communications Manager Release 6.x Network Services and HOST-RESOURCES-MIB

Cisco Unified CM Release 6.x Network Services	hrSWRunName OIDs	Clearing Values(Positive String)	Object Responses
Cisco AMC Service Service	1.3.6.1.2.1.25.4.2.1.2	amc	Cisco AMC Service Service Failure
Cisco CAR Scheduler Service		carschlr	Cisco CAR Scheduler Service Failure
Cisco Trace Collection Service		tracecollection	Cisco Trace Collection Service Failure
HOST-RESOURCES MIB activation on system		hostagt	Host MIB Query Capabilities Disabled
SYSAPPL-MIB activation on system	1.3.6.1.2.1.54	sappagt	SysApp MIB Query Capabilities Disabled
MIB2 (RFC1213) activation on system	1.3.6.1.2.1	mib2agt	MIB2 MIB Query Capabilities Disabled
SNMP activation on system	1.3.6.1.2.1.25.4.2.1.2	snmp_master_age	System SNMP Capabilities are Disabled
SNMP activation on system		snmpd	SNMP Capabilities are Disabled
Native Agent Adaptor activation on system		naaagt	Native Adaptor Agent Capabilities are Disabled
RIS Data Collector Service		RisDC	RIS Data Collector Service Failure
CDR Agent Service		cdragent	CDR Agent Service Failure
CDR Replication Service		cdrrep	CDR Replication Service Failure
Database Layer Replication Service		dblrpc	Database Layer Replication Service Failure
Database Layer Monitor Service		dbmon	Database Layer Monitor Service Failure
SSH Service		sshd	SSH Service Failure
Syslog Service		syslogd	Syslog Service Failure
License Manager Service		CiscoLicenseMgr	License Manager Service Failure

Cisco Unified CM Release 6.x Network Services	hrSWRunName OIDs	Clearing Values(Positive String)	Object Responses
System Backup Master Service		CiscoDRFMaster	System Backup Master Service Failure
System Backup Local Service		CiscoDRFLocal	System Backup Local Service Failure
CISCO-CDP-MIB activation on system	1.3.6.1.4.1.9.9.23	cdpAgt	CDP MIB Query Capabilities Disabled
CDP service		cdpd	CDP Service Failure
Certificate Expiry Monitor Service Availability	1.3.6.1.2.1.25.4.2.1.2	certM	Certificate Expiry Monitor Service Failure
Syslog Service		CiscoSyslogSubA	Syslog Service Failure
Database Service		cmoninit	
HOST-RESOURCES MIB activation on system	1.3.6.1.2.1.25	host_agent.pl	Host MIB Query Capabilities Disabled
Tomcat Service		tomcat	Tomcat Service Failure
Log Partition Monitoring Tool Service		LpmTool	Log Partition Monitoring Tool Service Failure
SNMP activation on system		snmpdm	System SNMP Capabilities are Disabled

Troubleshoot Host Resources MIB

The following logs and information needs to be collected for troubleshooting purpose:

- The hostagt log files by executing the **file get activelog** `/platform/snmp/hostagt/` command.
- The syslog files by executing the **file get activelog** `/syslog/` command.
- Master SNMP Agent log files by executing the **file get activelog** `/platform/snmp/snmpdm/` command.
- Sequence of operations performed.

Frequently Asked Questions for Host Resources MIB

Q. Can the HOST-RESOURCES-MIB be used for process monitoring?

A. Host resources MIB does retrieve the information about the processes running on the system in hrSwRunTable. But this monitors all the processes running in the system. If you need to monitor only the installed Cisco Application, then the best way is to use SYSAPPL-MIB.

Q. How is the memory usage values shown by RTMT mapped to the HOST-RESOURCES-MIB?

A. The following table lists the memory usage values.

Table 5: Memory Usage Values

Memory Usages	RTMT Counter	HOST-RESOURCES-MIB
SWAP memory Usage	Memory\Used Swap Kbytes	hrStorageUsed.2 (whose description is Virtual Memory)
Physical Memory Usage	Memory\Used Kbytes	hrStorageUsed.1(whose description is Physical RAM)
Total memory (physical + swap) usage	Memory\Used VM Kbytes	No equivalent. Basically need to add hrStorageUsed.2 and hrStorageUsed.1 Since swap memory may not be used at all on lightly used servers, HR Virtual Memory may return 0. To validate HR VM is returning correctly, that value needs to be compared against RTMT Memory\Used Swap KBytes. It's unfortunate that RTMT and HR use the term "Virtual memory" differently but that's what we have to work with. The hrStorageUsed for physical memory shows the data in terms of used - (buffers + cache).

Memory Usages	RTMT Counter	HOST-RESOURCES-MIB
		The hrStorageUsed for physical memory shows the data in terms of used that is buffers + cache. The shared memory info that is exposed by the MIB is HOST-RESOURCES-MIB::hrStorageDescr.10 = STRING: /dev/shm.The virtual memory reported by HOST-RESOURCES-MIB is what is considered as swap memory by RTMT. For HOST RESOURCES MIB, the following is used: • %Physical memory usage = (Physical RAM hrStorageUsed + /dev/shm hrStorageUsed) / (Physical RAM hrStorageSize) • %VM used = (Physical RAM hrStorageUsed + /dev/shm hrStorageUsed + Virtual Memory hrStorageUsed) / (Physical RAM hrStorageSize + Virtual Memory hrStorageSize)

Q. Why do the disk space values shown by RTMT and the HOST-RESOURCES-MIB differ?

A. In general the df size will not match the used and available disk space data shown. This is because of minfree percentage of reserved filesystem disk blocks. The minfree value for a Unified Communications Manager in Releases 6.x and 7.0 systems is 1%. So there will be difference of 1% between the disk space used value shown in RTMT and HOST-RESOURCES-MIB.

In RTMT, the disk space used value is shown from df reported values: [(Total Space - Available Space) /Total Space] * 100 where the Total Space includes the minfree also. For the HOST-RESOURCES-MIB, this is calculated by [hrStorageUsed/hrStorageSize] * 100 wherein the hrStorageSize does not include the minfree.

Q. How does the Host Agent display the value in hrStorageUsed?

A. The hrStorageUsed for physical RAM was corrected to show the data in terms of used (buffers + cache). To check if the host agent version is correct, collect the snmp-rpm version installed in the system by using the show packages active snmp command.

How the memory capacity/usage values compare to those of HOST-RESOURCES-MIB?

In the HOST-RESOURCES-MIB the size and storage used are represented in terms of hrStorageUnits. If for that storage type, the hrStorageUnits is 4096 bytes then the hrStorageUsed or hrStorageSize value

queried in the MIB value should be multiplied by 4096. For example, the show status command displays the Total Memory as 4090068K for Physical RAM.

If hrStorageUnits for physicalRAM storage type is 4096 bytes, then hrStorageSize for Physical RAM will be shown as 1022517 which is 4090078K [(1022517 x 4096)/1024 = 4090068K].

Q. An SNMP query on hrSWRunName in HOST-RESOURCES-MIB intermittently returns incorrect entries in Windows.

A. The Microsoft SNMP extension agent (hostmib.dll) supports the HOST-RESOURCE-MIB. So Microsoft support may be able to help on this. If the problem is persistent then following is recommended:

- Use the tlist snmp.exe file to verify the hostmib.dll is listed in the output.
- Verify there are no error/warning messages from SNMP, in the event viewer, when SNMP service is started.
- Make sure the community string used has been configured with read privilege under snmp service properties.
- Use MSSQL-MIB (MssqlSrvInfoTable) to confirm sql process status

Q. Monitoring Processes

A. HOST-RESOURCES-MIB retrieves information about all the processes that are running on the system from hrSWRunTable. Use this MIB for monitoring all the processes that are running in the system. To monitor the only the installed Cisco application, use SYSAPPL-MIB.Disk Space and RTMT

The used and available disk space values that are shown by HOST-RESOURCES-MIB may not match the disk space values that are shown by RTMT due to the minfree percentage of reserved file system disk blocks. Because the minfree value for Unified Communications Manager in 6.x and 7.0 systems is 1 percent, you will see a 1 percent difference between the used disk space value that is shown by RTMT and HOST-RESOURCES-MIB.

- In RTMT, the disk space used value gets shown from df reported values: [(Total Space – Available Space) /Total Space] * 100 where the Total Space includes the minfree also.
- For Host Resources MIB, the disk space used value gets calculated by [hrStorageUsed/hrStorageSize] * 100 where the hrStorageSize does not include the minfree.

IF-MIB

Note This is a reformatted version of IF-MIB. Download and compile all of the MIBs in this section from http://tools.cisco.com/Support/SNMP/do/BrowseMIB.do?local=en&step=2.

Before you can compile IF-MIB, you need to compile the MIBs listed below in the order listed.

1. SNMPv2-SMI
2. SNMPv2-TC
3. SNMPv2-CONF

4. SNMPv2-MIB
5. IANAifType-MIB
6. RFC1155-SMI
7. RFC-1212
8. SNMPv2-SMI-v1
9. RFC-1215
10. SNMPv2-TC-v1
11. IF-MIB

Additional downloads are:

- OID File: IF-MIB.oid

IF-MIB Revisions

The following table lists the revisions to this MIB beginning with the latest revision.

Table 6: History of Revisions

Date	Action	Description
06/14/2000	Updated	The MIB module to describe generic objects for network interface sub-layers. This MIB is an updated version of MIB-II ifTable, and incorporates the extensions defined in RFC 1229. Clarifications agreed upon by the Interfaces MIB WG, and published as RFC 2863.
02/28/1996	Revised	Revisions made by the Interfaces MIB WG, and published in RFC 2233.
08/11/1993	Initial Version	Published as part of RFC 1573. ::= {mib-2 31}

IF-MIB Definitions

The following definitions are imported for IF-MIB:

- MODULE-IDENTITY, OBJECT-TYPE, Counter32, Gauge32, Counter64, Integer32, TimeTicks, mib-2, NOTIFICATION-TYPE

- From SNMPv2-SMI—TEXTUAL-CONVENTION, DisplayString, PhysAddress, TruthValue, RowStatus, TimeStamp, AutonomousType, TestAndIncr
- From SNMPv2-TC—MODULE-COMPLIANCE, OBJECT-GROUP, NOTIFICATION-GROUP
- From SNMPv2-CONF— snmpTraps
- From SNMPv2-MIB—IANAifType
- From IANAifType-MIB;

IF-MIB Objects

ifMIBObjects OBJECT IDENTIFIER

::= {ifMIB 1}

interfaces OBJECT IDENTIFIER

::= {mib-2 2}

IF-MIB Textual Conventions

Note OwnerString has the same semantics as used in RFC 1271.

OwnerString ::= TEXTUAL-CONVENTION

DISPLAY-HINT 255a

STATUS deprecated

DESCRIPTION

This data type is used to model an administratively assigned name of the owner of a resource. This information is taken from the NVT ASCII character set. It is suggested that this name contain one or more of the following: ASCII form of the manager station's transport address, management station name (e.g., domain name), network management personnel's name, location, or phone number. In some cases the agent itself will be the owner of an entry. In these cases, this string shall be set to a string starting with agent.

A value which indicates the set of services that this entity may potentially offers. The value is a sum. This sum initially takes the value zero, Then, for each layer, L, in the range 1 through 7, that this node performs transactions for, 2 raised to (L - 1) is added to the sum. For example, a node which performs only routing functions would have a value of 4 (2^(3-1)). In contrast, a node which is a host offering application services would have a value of 72 (2^(4-1) + 2^(7-1)). Note that in the context of the Internet suite of protocols, values should be calculated accordingly:

Layer functionality:

- 1—physical (e.g., repeaters)
- 2—datalink/subnetwork (e.g., bridges)
- 3—internet (e.g., supports the IP)

- 4—end-to-end (e.g., supports the TCP)
- 7—applications (e.g., supports the SMTP)

  For systems including OSI protocols, layers 5 and 6 may also be counted.

  SYNTAX Octet String (SIZE(0..255))

Interface Index

The Interface Index contains the semantics of ifIndex and should be used for any objects defined in other MIB modules that need these semantics.

InterfaceIndex ::= TEXTUAL-CONVENTION

DISPLAY-HINT d

STATUS current

DESCRIPTION

A unique value, greater than zero, for each interface or interface sub-layer in the managed system. It is recommended that values are assigned contiguously starting from 1. The value for each interface sub-layer must remain constant at least from one re-initialization of the entity's network management system to the next re-initialization.

SYNTAX Integer32 (1..2147483647)

InterfaceIndexOrZero ::= TEXTUAL-CONVENTION

DISPLAY-HINT d

STATUS current

DESCRIPTION

This textual convention is an extension of the InterfaceIndex convention. The latter defines a greater than zero value used to identify an interface or interface sub-layer in the managed system. This extension permits the additional value of zero. The value zero is object-specific and must therefore be defined as part of the description of any object which uses this syntax. Examples of the usage of zero might include situations where interface was unknown, or when none or all interfaces need to be referenced.

SYNTAX Integer32 (0..2147483647)

ifNumber OBJECT-TYPE

SYNTAX Integer32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The number of network interfaces (regardless of their current state) present on this system.

::= { interfaces 1 }

ifTableLastChange OBJECT-TYPE

SYNTAX TimeTicks

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The value of sysUpTime at the time of the last creation or deletion of an entry in the ifTable. If the number of entries has been unchanged since the last re-initialization of the local network management subsystem, then this object contains a zero value.

::= {ifMIBObjects 5}

Interfaces Table

The Interfaces table contains information on the entity's interfaces. Each sub-layer below the internetwork-layer of a network interface is considered to be an interface.

ifTable OBJECT-TYPE

SYNTAX Sequence of IfEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

A list of interface entries. The number of entries is given by the value of ifNumber.

::= {interfaces 2}

ifEntry OBJECT-TYPE

SYNTAX IfEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

An entry containing management information applicable to a particular interface.

INDEX {ifIndex}

::= {ifTable 1}

IfEntry ::=

SEQUENCE {ifIndex InterfaceIndex, ifDescr DisplayString, ifType IANAifType, ifMtu Integer32, filespec Gauge32, ifPhysAddress PhysAddress, ifAdminStatus INTEGER, ifOperStatusINTEGER, ifLastChangeTimeTicks, ifInOctets Counter32, ifInUcastPkts Counter32, ifInNUcastPkts Counter32, -- deprecated ifInDiscardsCounter32, ifInErrors Counter32, ifInUnknownProtos Counter32, ifOutOctets Counter32, ifOutUcastPkts Counter32, ifOutNUcastPkts Counter32, -- deprecated ifOutDiscards Counter32, ifOutErrors Counter32, ifOutQLen Gauge32,-- deprecated ifSpecific OBJECT IDENTIFIER -- deprecated}

ifIndex OBJECT-TYPE

SYNTAX InterfaceIndex

MAX-ACCESS read-only

STATUS current

DESCRIPTION

A unique value, greater than zero, for each interface. It is recommended that values are assigned contiguously starting from 1. The value for each interface sub-layer must remain constant at least from one re-initialization of the entity's network management system to the next re-initialization.

::= {ifEntry 1}

ifDescr OBJECT-TYPE

SYNTAX DisplayString (SIZE (0..255))

MAX-ACCESS read-only

STATUS current

DESCRIPTION

A textual string containing information about the interface. This string should include the name of the manufacturer, the product name and the version of the interface hardware/software.

::= {ifEntry 2}

ifType OBJECT-TYPE

SYNTAX IANAifType

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The type of interface. Additional values for ifType are assigned by the Internet Assigned Numbers Authority (IANA), through updating the syntax of the IANAifType textual convention.

::= {ifEntry 3}

ifMtu OBJECT-TYPE

SYNTAX Integer32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The size of the largest packet which can be sent/received on the interface, specified in octets. For interfaces that are used for transmitting network datagrams, this is the size of the largest network datagram that can be sent on the interface.

::= {ifEntry 4}

ifSpeed OBJECT-TYPE

SYNTAX Gauge32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

An estimate of the interface current bandwidth in bits per second. For interfaces which do not vary in bandwidth or for those where no accurate estimation can be made, this object should contain the nominal

bandwidth. If the bandwidth of the interface is greater than the maximum value reportable by this object then this object should report its maximum value (4,294,967,295) and ifHighSpeed must be used to report the interface speed. For a sub-layer which has no concept of bandwidth, this object should be zero.

::= {ifEntry 5}

ifPhysAddress OBJECT-TYPE

SYNTAX PhysAddress

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The interface's address at its protocol sub-layer. For example, for an 802.x interface, this object normally contains a MAC address. The interface's media-specific MIB must define the bit and byte ordering and the format of the value of this object. For interfaces which do not have such an address (e.g., a serial line), this object should contain an octet string of zero length.

::= {ifEntry 6}

ifAdminStatus OBJECT-TYPE

SYNTAX Integer {up(1), -- ready to pass packets down(2), testing(3) -- in some test mode}

MAX-ACCESS read-write

STATUS current

DESCRIPTION

The desired state of the interface. The testing(3) state indicates that no operational packets can be passed. When a managed system initializes, all interfaces start with ifAdminStatus in the down(2) state. As a result of either explicit management action or per configuration information retained by the managed system, ifAdminStatus is then changed to either the up(1) or testing(3) states (or remains in the down(2) state).

::= {if Entry 7}

ifOperStatus OBJECT-TYPE

SYNTAX INTEGER {up(1),-- ready to pass packets down(2), testing(3), -- in some test mode unknown(4), -- status can not be determined -- for some reason. dormant(5), notPresent(6),-- some component is missing lowerLayerDown(7) -- down due to state of -- lower-layer interface(s)}

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The current operational state of the interface. The testing(3) state indicates that no operational packets can be passed. If ifAdminStatus is down(2) then ifOperStatus should be down(2). If ifAdminStatus is changed to up(1) then ifOperStatus should change to up(1) if the interface is ready to transmit and receive network traffic; it should change to dormant(5) if the interface is waiting for external actions (such as a serial line waiting for an incoming connection); it should remain in the down(2) state if and only if there is a fault that prevents it from going to the up(1) state; it should remain in the notPresent(6) state if the interface has missing (typically, hardware) components.

::= {ifEntry 8}

ifLastChange OBJECT-TYPE

SYNTAX TimeTicks

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The value of sysUpTime at the time the interface entered its current operational state. If the current state was entered prior to the last re-initialization of the local network management subsystem, then this object contains a zero value.

::= {ifEntry 9}

ifInOctets OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The total number of octets received on the interface, including framing characters. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.

::= { ifEntry 10 }

ifInUcastPkts OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were not addressed to a multicast or broadcast address at this sub-layer. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.

::= { ifEntry 11 }

ifInNUcastPkts OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS deprecated

DESCRIPTION

The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were addressed to a multicast or broadcast address at this sub-layer. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.

This object is deprecated in favour of ifInMulticastPkts and ifInBroadcastPkts.

::= { ifEntry 12 }

ifInDiscards OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. One possible reason for discarding such a packet could be to free up buffer space. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.

::= { ifEntry 13 }

ifInErrors OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

For packet-oriented interfaces, the number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol. For character-oriented or fixed-length interfaces, the number of inbound transmission units that contained errors preventing them from being deliverable to a higher-layer protocol.

Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.

::= { ifEntry 14 }

ifInUnknownProtos OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

For packet-oriented interfaces, the number of packets received via the interface which were discarded because of an unknown or unsupported protocol. For character-oriented or fixed-length interfaces that support protocol multiplexing the number of transmission units received via the interface which were discarded because of an unknown or unsupported protocol. For any interface that does not support protocol multiplexing, this counter will always be 0.

Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.

::= { ifEntry 15 }

ifOutOctets OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The total number of octets transmitted out of the interface, including framing characters. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.

::= { ifEntry 16 }

ifOutUcastPkts OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The total number of packets that higher-level protocols requested be transmitted, and which were not addressed to a multicast or broadcast address at this sub-layer, including those that were discarded or not sent. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.

::= { ifEntry 17 }

ifOutNUcastPkts OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS deprecated

DESCRIPTION

The total number of packets that higher-level protocols requested be transmitted, and which were addressed to a multicast or broadcast address at this sub-layer, including those that were discarded or not sent. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.

This object is deprecated in favour of ifOutMulticastPkts and ifOutBroadcastPkts.

::= { ifEntry 18 }

ifOutDiscards OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.

::= { ifEntry 19 }

ifOutErrors OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

For packet-oriented interfaces, the number of outbound packets that could not be transmitted because of errors. For character-oriented or fixed-length interfaces, the number of outbound transmission units that could not be transmitted because of errors. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.

::= { ifEntry 20 }

ifOutQLen OBJECT-TYPE

SYNTAX Gauge32

MAX-ACCESS read-only

STATUS deprecated

DESCRIPTION

The length of the output packet queue (in packets).

::= { ifEntry 21 }

ifSpecific OBJECT-TYPE

SYNTAX OBJECT IDENTIFIER

MAX-ACCESS read-only

STATUS deprecated

DESCRIPTION

A reference to MIB definitions specific to the particular media being used to realize the interface. It is recommended that this value point to an instance of a MIB object in the media-specific MIB, i.e., that this object have the semantics associated with the InstancePointer textual convention defined in RFC 2579. In fact, it is recommended that the media-specific MIB specify what value ifSpecific should/can take for values of ifType. If no MIB definitions specific to the particular media are available, the value should be set to the OBJECT IDENTIFIER { 0 0 }.

::= { ifEntry 22 }

Extension to the Interface Table

This table replaces the ifExtnsTable table.

ifXTableOBJECT-TYPE

SYNTAX Sequence of IfXEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

A list of interface entries. The number of entries is given by the value of ifNumber. This table contains additional objects for the interface table.

::= { ifMIBObjects 1 }

ifXEntryOBJECT-TYPE

SYNTAX IfXEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

An entry containing additional management information applicable to a particular interface.

AUGMENTS{ ifEntry }

::= { ifXTable 1 }

IfXEntry ::=

SEQUENCE {ifName DisplayString, ifInMulticastPkts Counter32, ifInBroadcastPkts Counter32, ifOutMulticastPkts Counter32, ifOutBroadcastPkts Counter32, ifHCInOctetsCounter64, ifHCInUcastPkts Counter64, ifHCInMulticastPkts Counter64, ifHCInBroadcastPkts Counter64, ifHCOutOctets Counter64, ifHCOutUcastPktsCounter64, ifHCOutMulticastPktsCounter64, ifHCOutBroadcastPktsCounter64, ifLinkUpDownTrapEnable INTEGER, ifHighSpeed Gauge32, ifPromiscuousMode TruthValue, ifConnectorPresent TruthValue, ifAlias DisplayString, ifCounterDiscontinuityTime TimeStamp }

ifName OBJECT-TYPE

SYNTAX DisplayString

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The textual name of the interface. The value of this object should be the name of the interface as assigned by the local device and should be suitable for use in commands entered at the device's 'console'. This might be a text name, such as 'le0' or a simple port number, such as '1', depending on the interface naming syntax of the device. If several entries in the ifTable together represent a single interface as named by the device, then each will have the same value of ifName. Note that for an agent which responds to SNMP queries concerning an interface on some other (proxied) device, then the value of ifName for such an interface is the proxied device's local name for it.

If there is no local name, or this object is otherwise not applicable, then this object contains a zero-length string.

::= { ifXEntry 1 }

ifInMulticastPkts OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were addressed to a multicast address at this sub-layer. For a MAC layer protocol, this includes both Group and Functional addresses. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.

::= { ifXEntry 2 }

ifInBroadcastPkts OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were addressed to a broadcast address at this sub-layer. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.

::= { ifXEntry 3 }

ifOutMulticastPkts OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The total number of packets that higher-level protocols requested be transmitted, and which were addressed to a multicast address at this sub-layer, including those that were discarded or not sent. For a MAC layer protocol, this includes both Group and Functional addresses.

Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.

::= { ifXEntry 4 }

ifOutBroadcastPkts OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The total number of packets that higher-level protocols requested be transmitted, and which were addressed to a broadcast address at this sub-layer, including those that were discarded or not sent.

Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.

::= { ifXEntry 5 }

High Capacity Counter Objects

These objects are all 64 bit versions of the basic ifTable counters. These objects all have the same basic semantics as their 32-bit counterparts, however, their syntax has been extended to 64 bits.

ifHCInOctets OBJECT-TYPE

SYNTAX Counter64

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The total number of octets received on the interface, including framing characters. This object is a 64-bit version of ifInOctets. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.

::= { ifXEntry 6 }

ifHCInUcastPkts OBJECT-TYPE

SYNTAX Counter64

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were not addressed to a multicast or broadcast address at this sub-layer. This object is a 64-bit version of ifInUcastPkts.

Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.

::= { ifXEntry 7 }

ifHCInMulticastPkts OBJECT-TYPE

SYNTAX Counter64

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were addressed to a multicast address at this sub-layer. For a MAC layer protocol, this includes both Group and Functional addresses. This object is a 64-bit version of ifInMulticastPkts.

Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.

::= { ifXEntry 8 }

ifHCInBroadcastPkts OBJECT-TYPE

SYNTAX Counter64

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were addressed to a broadcast address at this sub-layer. This object is a 64-bit version of ifInBroadcastPkts.

Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.

::= { ifXEntry 9 }

ifHCOutOctets OBJECT-TYPE

SYNTAX Counter64

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The total number of octets transmitted out of the interface, including framing characters. This object is a 64-bit version of ifOutOctets.

Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.

::= { ifXEntry 10 }

ifHCOutUcastPkts OBJECT-TYPE

SYNTAX Counter64

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The total number of packets that higher-level protocols requested be transmitted, and which were not addressed to a multicast or broadcast address at this sub-layer, including those that were discarded or not sent. This object is a 64-bit version of ifOutUcastPkts.

Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.

::= { ifXEntry 11 }

ifHCOutMulticastPkts OBJECT-TYPE

SYNTAX Counter64

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The total number of packets that higher-level protocols requested be transmitted, and which were addressed to a multicast address at this sub-layer, including those that were discarded or not sent. For a MAC layer protocol, this includes both Group and Functional addresses. This object is a 64-bit version of ifOutMulticastPkts.

Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.

::= { ifXEntry 12 }

ifHCOutBroadcastPkts OBJECT-TYPE

SYNTAX Counter64

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The total number of packets that higher-level protocols requested be transmitted, and which were addressed to a broadcast address at this sub-layer, including those that were discarded or not sent. This object is a 64-bit version of ifOutBroadcastPkts.

Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime.

::= { ifXEntry 13 }

ifLinkUpDownTrapEnable OBJECT-TYPE

SYNTAX Integer { enabled(1), disabled(2) }

MAX-ACCESS read-write

STATUS current

DESCRIPTION

Indicates whether linkUp/linkDown traps should be generated for this interface. By default, this object should have the value enabled(1) for interfaces which do not operate on 'top' of any other interface (as defined in the ifStackTable), and disabled(2) otherwise.

::= { ifXEntry 14 }

ifHighSpeed OBJECT-TYPE

SYNTAX Gauge32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

An estimate of the interface's current bandwidth in units of 1,000,000 bits per second. If this object reports a value of 'n' then the speed of the interface is somewhere in the range of 'n-500,000' to 'n+499,999'. For interfaces which do not vary in bandwidth or for those where no accurate estimation can be made, this object should contain the nominal bandwidth. For a sub-layer which has no concept of bandwidth, this object should be zero.

::= { ifXEntry 15 }

ifPromiscuousMode OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-write

STATUS current

DESCRIPTION

This object has a value of false(2) if this interface only accepts packets/frames that are addressed to this station. This object has a value of true(1) when the station accepts all packets/frames transmitted on the media. The value true(1) is only legal on certain types of media. If legal, setting this object to a value of true(1) may require the interface to be reset before becoming effective.

The value of ifPromiscuousMode does not affect the reception of broadcast and multicast packets/frames by the interface.

::= { ifXEntry 16 }

ifConnectorPresent OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-only

STATUS current

DESCRIPTION

This object has the value 'true(1)' if the interface sublayer has a physical connector and the value 'false(2)' otherwise.

::= { ifXEntry 17 }

ifAlias OBJECT-TYPE

SYNTAX DisplayString (SIZE(0..64))

MAX-ACCESS read-write

STATUS current

DESCRIPTION

This object is an alias name for the interface as specified by a network manager, and provides a non-volatile handle for the interface.

On the first instantiation of an interface, the value of ifAlias associated with that interface is the zero-length string. As and when a value is written into an instance of ifAlias through a network management set operation, then the agent must retain the supplied value in the ifAlias instance associated with the same interface for as long as that interface remains instantiated, including across all re-initializations/reboots of the network management system, including those which result in a change of the interface's ifIndex value.

An example of the value which a network manager might store in this object for a WAN interface is the (Telco's) circuit number/identifier of the interface.

Some agents may support write-access only for interfaces having particular values of ifType. An agent which supports write access to this object is required to keep the value in non-volatile storage, but it may limit the length of new values depending on how much storage is already occupied by the current values for other interfaces.

::= { ifXEntry 18 }

ifCounterDiscontinuityTime OBJECT-TYPE

SYNTAX TimeStamp

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The value of sysUpTime on the most recent occasion at which any one or more of this interface's counters suffered a discontinuity. The relevant counters are the specific instances associated with this interface of any Counter32 or Counter64 object contained in the ifTable or ifXTable. If no such discontinuities have occurred since the last re-initialization of the local management subsystem, then this object contains a zero value.

::= { ifXEntry 19 }

Interface Stack Group

Implementation of this group is optional, but strongly recommended for all systems.

ifStackTable OBJECT-TYPE

SYNTAX Sequence of IfStackEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

The table containing information on the relationships between the multiple sub-layers of network interfaces. In particular, it contains information on which sub-layers run 'on top of' which other sub-layers, where each sub-layer corresponds to a conceptual row in the ifTable. For example, when the sub-layer with ifIndex value x runs over the sub-layer with ifIndex value y, then this table contains ifStackStatus.x.y=active.

For each ifIndex value, I, which identifies an active interface, there are always at least two instantiated rows in this table associated with I. For one of these rows, I is the value of ifStackHigherLayer; for the other, I is the value of ifStackLowerLayer. (If I is not involved in multiplexing, then these are the only two rows associated with I.)

For example, two rows exist even for an interface which has no others stacked on top or below it:

- ifStackStatus.0.x=active
- ifStackStatus.x.0=active

::= { ifMIBObjects 2 }

ifStackEntry OBJECT-TYPE

SYNTAX IfStackEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

Information on a particular relationship between two sub-layers, specifying that one sub-layer runs on 'top' of the other sub-layer. Each sub-layer corresponds to a conceptual row in the ifTable.

INDEX { ifStackHigherLayer, ifStackLowerLayer }

::= { ifStackTable 1 }

IfStackEntry ::= SEQUENCE { ifStackHigherLayer InterfaceIndexOrZero, ifStackLowerLayer InterfaceIndexOrZero, ifStackStatus RowStatus }

ifStackHigherLayer OBJECT-TYPE

SYNTAX InterfaceIndexOrZero

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

The value of ifIndex corresponding to the higher sub-layer of the relationship, i.e., the sub-layer which runs on 'top' of the sub-layer identified by the corresponding instance of ifStackLowerLayer. If there is no higher sub-layer (below the internetwork layer), then this object has the value 0.

::= { ifStackEntry 1 }

ifStackLowerLayer OBJECT-TYPE

SYNTAX InterfaceIndexOrZero

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

The value of ifIndex corresponding to the lower sub-layer of the relationship, i.e., the sub-layer which runs 'below' the sub-layer identified by the corresponding instance of ifStackHigherLayer. If there is no lower sub-layer, then this object has the value 0.

::= { ifStackEntry 2 }

ifStackStatus OBJECT-TYPE

SYNTAX RowStatus

MAX-ACCESS read-create

STATUS current

DESCRIPTION

The status of the relationship between two sub-layers. Changing the value of this object from 'active' to 'notInService' or 'destroy' will likely have consequences up and down the interface stack. Thus, write access to this object is likely to be inappropriate for some types of interfaces, and many implementations will choose not to support write-access for any type of interface.

::= { ifStackEntry 3 }

ifStackLastChange OBJECT-TYPE

SYNTAX TimeTicks

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The value of sysUpTime at the time of the last change of the (whole) interface stack. A change of the interface stack is defined to be any creation, deletion, or change in value of any instance of ifStackStatus.

If the interface stack has been unchanged since the last re-initialization of the local network management subsystem, then this object contains a zero value.

::= { ifMIBObjects 6 }

Generic Receive Address Table

This group of objects is mandatory for all types of interfaces which can receive packets/frames addressed to more than one address. This table replaces the ifExtnsRcvAddr table. The main difference is that this table makes use of the RowStatus textual convention, while ifExtnsRcvAddr did not.

ifRcvAddressTable OBJECT-TYPE

SYNTAX Sequence of IfRcvAddressEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

This table contains an entry for each address (broadcast, multicast, or uni-cast) for which the system will receive packets/frames on a particular interface, except as follows:

- For an interface operating in promiscuous mode, entries are only required for those addresses for which the system would receive frames were it not operating in promiscuous mode.
- For 802.5 functional addresses, only one entry is required, for the address which has the functional address bit ANDed with the bit mask of all functional addresses for which the interface will accept frames.

  A system is normally able to use any unicast address which corresponds to an entry in this table as a source address.

::= { ifMIBObjects 4 }

ifRcvAddressEntry OBJECT-TYPE

SYNTAX IfRcvAddressEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

A list of objects identifying an address for which the system will accept packets/frames on the particular interface identified by the index value ifIndex.

INDEX { ifIndex, ifRcvAddressAddress }

::= { ifRcvAddressTable 1 }

IfRcvAddressEntry ::= SEQUENCE { ifRcvAddressAddress PhysAddress, ifRcvAddressStatusRowStatus, ifRcvAddressType INTEGER }

ifRcvAddressAddress OBJECT-TYPE

SYNTAX PhysAddress

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

An address for which the system will accept packets/frames on this entry's interface.

::= { ifRcvAddressEntry 1 }

ifRcvAddressStatus OBJECT-TYPE

SYNTAX RowStatus

MAX-ACCESS read-create

STATUS current

DESCRIPTION

This object is used to create and delete rows in the ifRcvAddressTable.

::= { ifRcvAddressEntry 2 }

ifRcvAddressType OBJECT-TYPE

SYNTAX INTEGER {other(1), volatile(2), nonVolatile(3) }

MAX-ACCESS read-create

STATUS current

DESCRIPTION

This object has the value nonVolatile(3) for those entries in the table which are valid and will not be deleted by the next restart of the managed system. Entries having the value volatile(2) are valid and exist, but have not been saved, so that will not exist after the next restart of the managed system. Entries having the value other(1) are valid and exist but are not classified as to whether they will continue to exist after the next restart.

DEFVAL { volatile }

::= { ifRcvAddressEntry 3 }

Definition of Interface-Related Traps

linkDown NOTIFICATION-TYPE

OBJECTS { ifIndex, ifAdminStatus, ifOperStatus }

STATUS current

DESCRIPTION

A linkDown trap signifies that the SNMP entity, acting in an agent role, has detected that the ifOperStatus object for one of its communication links is about to enter the down state from some other state (but not from the notPresent state). This other state is indicated by the included value

of ifOperStatus.

::= { snmpTraps 3 }

linkUp NOTIFICATION-TYPE

OBJECTS { ifIndex, ifAdminStatus, ifOperStatus }

STATUS current

DESCRIPTION

A linkUp trap signifies that the SNMP entity, acting in an agent role, has detected that the ifOperStatus object for one of its communication links left the down state and transitioned into some other state (but not into the notPresent state). This other state is indicated by the included value of ifOperStatus.

::= { snmpTraps 4 }

IF-MIB Conformance Information

ifConformance OBJECT IDENTIFIER

::= { ifMIB 2 }

ifGroups OBJECT IDENTIFIER

::= { ifConformance 1 }

ifCompliances OBJECT IDENTIFIER

::= { ifConformance 2 }

IF-MIB Compliance Statements

ifCompliance3 MODULE-COMPLIANCE

STATUS current

DESCRIPTION

The compliance statement for SNMP entities which have network interfaces.

MODULE -- this module

MANDATORY-GROUPS { ifGeneralInformationGroup, linkUpDownNotificationsGroup }

The groups:

- ifFixedLengthGroup
- ifHCFixedLengthGroup
- ifPacketGroup
- ifHCPacketGroup
- ifVHCPacketGroup

Mutually exclusive; at most one of these groups is implemented for a particular interface. When any of these groups is implemented for a particular interface, then ifCounterDiscontinuityGroup must also be implemented for that interface.

GROUP ifFixedLengthGroup

DESCRIPTION

This group is mandatory for those network interfaces which are character-oriented or transmit data in fixed-length transmission units, and for which the value of the corresponding instance of ifSpeed is less than or equal to 20,000,000 bits/second.

GROUP ifHCFixedLengthGroup

DESCRIPTION

This group is mandatory for those network interfaces which are character-oriented or transmit data in fixed-length transmission units, and for which the value of the corresponding instance of ifSpeed is greater than 20,000,000 bits/second.

GROUP ifPacketGroup

DESCRIPTION

This group is mandatory for those network interfaces which are packet-oriented, and for which the value of the corresponding instance of ifSpeed is less than or equal to 20,000,000 bits/second.

GROUP ifHCPacketGroup

DESCRIPTION

This group is mandatory only for those network interfaces which are packet-oriented and for which the value of the corresponding instance of ifSpeed is greater than 20,000,000 bits/second but less than or equal to 650,000,000 bits/second.

GROUP ifVHCPacketGroup

DESCRIPTION

This group is mandatory only for those network interfaces which are packet-oriented and for which the value of the corresponding instance of ifSpeed is greater than 650,000,000 bits/second.

GROUP ifCounterDiscontinuityGroup

DESCRIPTION

This group is mandatory for those network interfaces that are required to maintain counters (i.e., those for which one of the ifFixedLengthGroup, ifHCFixedLengthGroup, ifPacketGroup, ifHCPacketGroup, or ifVHCPacketGroup is mandatory).

GROUP ifRcvAddressGroup

DESCRIPTION

The applicability of this group MUST be defined by the media-specific MIBs. Media-specific MIBs must define the exact meaning, use, and semantics of the addresses in this group.

OBJECT ifLinkUpDownTrapEnable

MIN-ACCESS read-only

DESCRIPTION

Write access is not required.

OBJECT ifPromiscuousMode

MIN-ACCESS read-only

DESCRIPTION

Write access is not required.

OBJECT ifAdminStatus

SYNTAX INTEGER { up(1), down(2) }

MIN-ACCESS read-only

DESCRIPTION

Write access is not required, nor is support for the value testing(3).

OBJECT ifAlias

MIN-ACCESS read-only

DESCRIPTION

Write access is not required.

::= { ifCompliances 3 }

IF-MIB Units of Conformance

ifGeneralInformationGroupOBJECT-GROUP

OBJECTS { ifIndex, ifDescr, ifType, ifSpeed, ifPhysAddress, ifAdminStatus, ifOperStatus, ifLastChange, ifLinkUpDownTrapEnable, ifConnectorPresent, ifHighSpeed, ifName, ifNumber, ifAlias, ifTableLastChange }

STATUS current

DESCRIPTION

A collection of objects providing information applicable to all network interfaces.

::= { ifGroups 10 }

Note The following five groups are mutually exclusive; at most one of these groups is implemented for any interface.

- ifFixedLengthGroupOBJECT-GROUP

  OBJECTS { ifInOctets, ifOutOctets, ifInUnknownProtos, ifInErrors, ifOutErrors }

  STATUS current

  DESCRIPTION

  A collection of objects providing information specific to non-high speed (non-high speed interfaces transmit and receive at speeds less than or equal to 20,000,000 bits/second) character-oriented or fixed-length-transmission network interfaces.

  ::= { ifGroups 2 }

ifHCFixedLengthGroupOBJECT-GROUP

OBJECTS { ifHCInOctets, ifHCOutOctets, ifInOctets, ifOutOctets, ifInUnknownProtos, ifInErrors, ifOutErrors }

STATUS current

DESCRIPTION

A collection of objects providing information specific to high speed (greater than 20,000,000 bits/second) character-oriented or fixed-length-transmission network interfaces.

::= { ifGroups 3 }

ifPacketGroupOBJECT-GROUP

OBJECTS { ifInOctets, ifOutOctets, ifInUnknownProtos, ifInErrors, ifOutErrors, ifMtu, ifInUcastPkts, ifInMulticastPkts, ifInBroadcastPkts, ifInDiscards, ifOutUcastPkts, ifOutMulticastPkts, ifOutBroadcastPkts, ifOutDiscards, ifPromiscuousMode }

STATUS current

DESCRIPTION

A collection of objects providing information specific to non-high speed (non-high speed interfaces transmit and receive at speeds less than or equal to 20,000,000 bits/second) packet-oriented network interfaces.

::= { ifGroups 4 }

ifHCPacketGroupOBJECT-GROUP

OBJECTS { ifHCInOctets, ifHCOutOctets, ifInOctets, ifOutOctets, ifInUnknownProtos, ifInErrors, ifOutErrors, ifMtu, ifInUcastPkts, ifInMulticastPkts, ifInBroadcastPkts, ifInDiscards, ifOutUcastPkts, ifOutMulticastPkts, ifOutBroadcastPkts, ifOutDiscards, ifPromiscuousMode }

STATUS current

DESCRIPTION

A collection of objects providing information specific to high speed (greater than 20,000,000 bits/second but less than or equal to 650,000,000 bits/second) packet-oriented network interfaces.

::= { ifGroups 5 }

ifVHCPacketGroupOBJECT-GROUP

OBJECTS { ifHCInUcastPkts, ifHCInMulticastPkts, ifHCInBroadcastPkts, ifHCOutUcastPkts, ifHCOutMulticastPkts, ifHCOutBroadcastPkts, ifHCInOctets, ifHCOutOctets, ifInOctets, ifOutOctets, ifInUnknownProtos, ifInErrors, ifOutErrors, ifMtu, ifInUcastPkts, ifInMulticastPkts, ifInBroadcastPkts, ifInDiscards, ifOutUcastPkts, ifOutMulticastPkts, ifOutBroadcastPkts, ifOutDiscards, ifPromiscuousMode }

STATUS current

DESCRIPTION

A collection of objects providing information specific to higher speed (greater than 650,000,000 bits/second) packet-oriented network interfaces.

::= { ifGroups 6 }

ifRcvAddressGroupOBJECT-GROUP

OBJECTS { ifRcvAddressStatus, ifRcvAddressType }

STATUS current

DESCRIPTION

A collection of objects providing information on the multiple addresses which an interface receives.

::= { ifGroups 7 }

ifStackGroup2OBJECT-GROUP

OBJECTS { ifStackStatus, ifStackLastChange }

STATUS current

DESCRIPTION

A collection of objects providing information on the layering of MIB-II interfaces.

::= { ifGroups 11 }

ifCounterDiscontinuityGroup OBJECT-GROUP

OBJECTS { ifCounterDiscontinuityTime }

STATUS current

DESCRIPTION

A collection of objects providing information specific to interface counter discontinuities.

::= { ifGroups 13 }

linkUpDownNotificationsGroup NOTIFICATION-GROUP

NOTIFICATIONS { linkUp, linkDown }

STATUS current

DESCRIPTION

The notifications which indicate specific changes in the value of ifOperStatus.

::= { ifGroups 14 }

IF-MIB Deprecated Definitions - Objects

Interface Test Table

This group of objects is optional and deprecated. However, a media-specific MIB may make implementation of this group mandatory. This table replaces the ifExtnsTestTable.

ifTestTable OBJECT-TYPE

SYNTAX SEQUENCE OF IfTestEntry

MAX-ACCESS not-accessible

STATUS deprecated

DESCRIPTION

This table contains one entry per interface. It defines objects which allow a network manager to instruct an agent to test an interface for various faults. Tests for an interface are defined in the media-specific MIB for that interface. After invoking a test, the object ifTestResult can be read to determine the outcome. If an agent cannot perform the test, ifTestResult is set to so indicate. The object ifTestCode can be used to provide further test-specific or interface-specific (or even enterprise-specific) information concerning the outcome of the test. Only one test can be in progress on each interface at any one time. If one test is in progress when another test is invoked, the second test is rejected. Some agents may reject a test when a prior test is active on another interface.

Before starting a test, a manager-station must first obtain 'ownership' of the entry in the ifTestTable for the interface to be tested. This is accomplished with the ifTestId and ifTestStatus objects as follows:

```
try_again:
get (ifTestId, ifTestStatus)
while (ifTestStatus != notInUse)
/*
* Loop while a test is running or some other
* manager is configuring a test.
*/
short delay
get (ifTestId, ifTestStatus)
}
/*
* Is not being used right now -- let's compete
* to see who gets it.
*/
lock_value = ifTestId
if ( set(ifTestId = lock_value, ifTestStatus = inUse,
ifTestOwner = 'my-IP-address') == FAILURE)
/*
* Another manager got the ifTestEntry -- go
* try again
*/
goto try_again;
/*
* I have the lock
*/
set up any test parameters.
/*
* This starts the test
*/
set(ifTestType = test_to_run);
```

Wait for test completion by polling ifTestResult when test completes, agent sets ifTestResult agent also sets ifTestStatus = 'notInUse' retrieve any additional test results, and ifTestId if (ifTestId == lock_value+1) results are valid.

A manager station first retrieves the value of the appropriate ifTestId and ifTestStatus objects, periodically repeating the retrieval if necessary, until the value of ifTestStatus is 'notInUse'. The manager station then tries to set the same ifTestId object to the value it just retrieved, the same ifTestStatus object to 'inUse', and the corresponding ifTestOwner object to a value indicating itself. If the set operation succeeds then the manager has obtained ownership of the ifTestEntry, and the value of the ifTestId object is incremented by the agent (per the semantics of TestAndIncr). Failure of the set operation indicates that some other manager has obtained ownership of the ifTestEntry.

Once ownership is obtained, any test parameters can be setup, and then the test is initiated by setting ifTestType. On completion of the test, the agent sets ifTestStatus to 'notInUse'. Once this occurs, the manager can retrieve the results. In the (rare) event that the invocation of tests by two network managers were to overlap, then there would be a possibility that the first test's results might be overwritten by the second test's results prior to the first results being read. This unlikely circumstance can be detected by a network manager retrieving ifTestId at the same time as retrieving the test results, and ensuring that the results are for the desired request.

If ifTestType is not set within an abnormally long period of time after ownership is obtained, the agent should time-out the manager, and reset the value of the ifTestStatus object back to 'notInUse'. It is suggested that this time-out period be 5 minutes.

In general, a management station must not retransmit a request to invoke a test for which it does not receive a response; instead, it properly inspects an agent's MIB to determine if the invocation was successful. Only if the invocation was unsuccessful, is the invocation request retransmitted.

Some tests may require the interface to be taken off-line in order to execute them, or may even require the agent to reboot after completion of the test. In these circumstances, communication with the management station invoking the test may be lost until after completion of the test. An agent is not required to support such tests. However, if such tests are supported, then the agent should make every effort to transmit a response to the request which invoked the test prior to losing communication. When the agent is restored to normal service, the results of the test are properly made available in the appropriate objects.

Note that this requires that the ifIndex value assigned to an interface must be unchanged even if the test causes a reboot. An agent must reject any test for which it cannot, perhaps due to resource constraints, make available at least the minimum amount of information after that test completes.

::= { ifMIBObjects 3 }

ifTestEntry OBJECT-TYPE

SYNTAX IfTestEntry

MAX-ACCESS not-accessible

STATUS deprecated

DESCRIPTION

An entry containing objects for invoking tests on an interface.

AUGMENTS { ifEntry }

::= { ifTestTable 1 }

IfTestEntry ::=

SEQUENCE { ifTestId TestAndIncr, ifTestStatus INTEGER, ifTestType AutonomousType, ifTestResult INTEGER, ifTestCode OBJECT IDENTIFIER, ifTestOwnerOwnerString }

ifTestId OBJECT-TYPE

SYNTAX TestAndIncr

MAX-ACCESS read-write

STATUS deprecated

DESCRIPTION

This object identifies the current invocation of the interface's test.

::= { ifTestEntry 1 }

ifTestStatus OBJECT-TYPE

SYNTAX INTEGER { notInUse(1), inUse(2) }

MAX-ACCESS read-write

STATUS deprecated

DESCRIPTION

This object indicates whether or not some manager currently has the necessary 'ownership' required to invoke a test on this interface. A write to this object is only successful when it changes its value from 'notInUse(1)' to 'inUse(2)'. After completion of a test, the agent resets the value back to 'notInUse(1)'.

::= { ifTestEntry 2 }

ifTestType OBJECT-TYPE

SYNTAX AutonomousType

MAX-ACCESS read-write

STATUS deprecated

DESCRIPTION

A control variable used to start and stop operator-initiated interface tests. Most OBJECT IDENTIFIER values assigned to tests are defined elsewhere, in association with specific types of interface. However, this document assigns a value for a full-duplex loopback test, and defines the special meanings of the subject identifier:

noTest OBJECT IDENTIFIER ::= { 0 0 }

When the value noTest is written to this object, no action is taken unless a test is in progress, in which case the test is aborted. Writing any other value to this object is only valid when no test is currently in progress, in which case the indicated test is initiated.

When read, this object always returns the most recent value that ifTestType was set to. If it has not been set since the last initialization of the network management subsystem on the agent, a value of noTest is returned.

::= { ifTestEntry 3 }

ifTestResult OBJECT-TYPE

SYNTAX INTEGER { none(1), -- no test yet requested success(2), inProgress(3), notSupported(4), unAbleToRun(5), -- due to state of system aborted(6), failed(7) }

MAX-ACCESS read-only

STATUS deprecated

DESCRIPTION

This object contains the result of the most recently requested test, or the value none(1) if no tests have been requested since the last reset. Note that this facility provides no provision for saving the results of one test when starting another, as could be required if used by multiple managers concurrently.

::= { ifTestEntry 4 }

ifTestCode OBJECT-TYPE

SYNTAX OBJECT IDENTIFIER

MAX-ACCESS read-only

STATUS deprecated

DESCRIPTION

This object contains a code which contains more specific information on the test result, for example an error-code after a failed test. Error codes and other values this object may take are specific to the type of interface and/or test. The value may have the semantics of either the AutonomousType or InstancePointer textual conventions as defined in RFC 2579. The identifier is testCodeUnknown OBJECT IDENTIFIER ::= { 0 0 } and defined for use if no additional result code is available.

::= { ifTestEntry 5 }

ifTestOwner OBJECT-TYPE

SYNTAX OwnerString

MAX-ACCESS read-write

STATUS deprecated

DESCRIPTION

The entity which currently has the 'ownership' required to invoke a test on this interface.

::= { ifTestEntry 6 }

IF-MIB Deprecated Definitions - Groups

ifGeneralGroupOBJECT-GROUP

OBJECTS { ifDescr, ifType, ifSpeed, ifPhysAddress, ifAdminStatus, ifOperStatus, ifLastChange, ifLinkUpDownTrapEnable, ifConnectorPresent, ifHighSpeed, ifName }

STATUS deprecated

DESCRIPTION

A collection of objects deprecated in favour of ifGeneralInformationGroup.

::= { ifGroups 1 }

ifTestGroupOBJECT-GROUP

OBJECTS { ifTestId, ifTestStatus, ifTestType, ifTestResult, ifTestCode, ifTestOwner }

STATUS deprecated

DESCRIPTION

A collection of objects providing the ability to invoke tests on an interface.

::= { ifGroups 8 }

ifStackGroupOBJECT-GROUP

OBJECTS { ifStackStatus }

STATUS deprecated

DESCRIPTION

The previous collection of objects providing information on the layering of MIB-II interfaces.

::= { ifGroups 9 }

ifOldObjectsGroupOBJECT-GROUP

OBJECTS { ifInNUcastPkts, ifOutNUcastPkts, ifOutQLen, ifSpecific }

STATUS deprecated

DESCRIPTION

The collection of objects deprecated from the original MIB-II interfaces group.

::= { ifGroups 12 }

IF-MIB Deprecated Definitions - Compliance

ifCompliance MODULE-COMPLIANCE

STATUS deprecated

DESCRIPTION

A compliance statement defined in a previous version of this MIB module, for SNMP entities which have network interfaces.

MODULE -- this module

MANDATORY-GROUPS { ifGeneralGroup, ifStackGroup }

GROUP ifFixedLengthGroup

DESCRIPTION

This group is mandatory for all network interfaces which are character-oriented or transmit data in fixed-length transmission units.

GROUP ifHCFixedLengthGroup

DESCRIPTION

This group is mandatory only for those network interfaces which are character-oriented or transmit data in fixed-length transmission units, and for which the value of the corresponding instance of ifSpeed is greater than 20,000,000 bits/second.

GROUP ifPacketGroup

DESCRIPTION

This group is mandatory for all network interfaces which are packet-oriented.

GROUP ifHCPacketGroup

DESCRIPTION

This group is mandatory only for those network interfaces which are packet-oriented and for which the value of the corresponding instance of ifSpeed is greater than 650,000,000 bits/second.

GROUP ifTestGroup

DESCRIPTION

This group is optional. Media-specific MIBs which require interface tests are strongly encouraged to use this group for invoking tests and reporting results. A medium specific MIB which has mandatory tests may make implementation of this group mandatory.

GROUP ifRcvAddressGroup

DESCRIPTION

The applicability of this group MUST be defined by the media-specific MIBs. Media-specific MIBs must define the exact meaning, use, and semantics of the addresses in this group.

OBJECT ifLinkUpDownTrapEnable

MIN-ACCESS read-only

DESCRIPTION

Write access is not required.

OBJECT ifPromiscuousMode

MIN-ACCESS read-only

DESCRIPTION

Write access is not required.

OBJECT ifStackStatus

SYNTAX INTEGER { active(1) } -- subset of RowStatus

MIN-ACCESS read-only

DESCRIPTION

Write access is not required, and only one of the six enumerated values for the RowStatus textual convention need be supported, specifically: active(1).

OBJECT ifAdminStatus

SYNTAX INTEGER { up(1), down(2) }

MIN-ACCESS read-only

DESCRIPTION

Write access is not required, nor is support for the value testing(3).

::= { ifCompliances 1 }

ifCompliance2 MODULE-COMPLIANCE

STATUS deprecated

DESCRIPTION

A compliance statement defined in a previous version of this MIB module, for SNMP entities which have network interfaces.

MODULE -- this module

MANDATORY-GROUPS { ifGeneralInformationGroup, ifStackGroup2, ifCounterDiscontinuityGroup }

GROUP ifFixedLengthGroup

DESCRIPTION

This group is mandatory for all network interfaces which are character-oriented or transmit data in fixed-length transmission units.

GROUP ifHCFixedLengthGroup

DESCRIPTION

This group is mandatory only for those network interfaces which are character-oriented or transmit data in fixed-length transmission units, and for which the value of the corresponding instance of ifSpeed is greater than 20,000,000 bits/second.

GROUP ifPacketGroup

DESCRIPTION

This group is mandatory for all network interfaces which are packet-oriented.

GROUP ifHCPacketGroup

DESCRIPTION

This group is mandatory only for those network interfaces which are packet-oriented and for which the value of the corresponding instance of ifSpeed is greater than 650,000,000 bits/second.

GROUP ifRcvAddressGroup

DESCRIPTION

The applicability of this group MUST be defined by the media-specific MIBs. Media-specific MIBs must define the exact meaning, use, and semantics of the addresses in this group.

OBJECT ifLinkUpDownTrapEnable

MIN-ACCESS read-only

DESCRIPTION

Write access is not required.

OBJECT ifPromiscuousMode

MIN-ACCESS read-only

DESCRIPTION

Write access is not required.

OBJECT ifStackStatus

SYNTAX INTEGER { active(1) } -- subset of RowStatus

MIN-ACCESS read-only

DESCRIPTION

Write access is not required, and only one of the six enumerated values for the RowStatus textual convention need be supported, specifically: active(1).

OBJECT ifAdminStatus

SYNTAX INTEGER { up(1), down(2) }

MIN-ACCESS read-only

DESCRIPTION

Write access is not required, nor is support for the value testing(3).

OBJECT ifAlias

MIN-ACCESS read-only

DESCRIPTION

Write access is not required.

::= { ifCompliances 2 }