

Definitions of Managed Objects for Remote Ping, Traceroute, and
Lookup Operations

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2000). All Rights Reserved.

Abstract

This memo defines Management Information Bases (MIBs) for performing remote ping, traceroute and lookup operations at a remote host. When managing a network it is useful to be able to initiate and retrieve the results of ping or traceroute operations when performed at a remote host. A Lookup capability is defined in order to enable resolving of either an IP address to a DNS name or a DNS name to an IP address at a remote host.

Currently, there are several enterprise-specific MIBs for performing remote ping or traceroute operations. The purpose of this memo is to define a standards-based solution to enable interoperability.

Table of Contents

1.0	Introduction	2
2.0	The SNMP Network Management Framework	4
3.0	Structure of the MIBs	5
3.1	Ping MIB	6
3.1.1	pingMaxConcurrentRequests	6
3.1.2	pingCtlTable	6
3.1.3	pingResultsTable	7
3.1.4	pingProbeHistoryTable	7
3.2	Traceroute MIB	8
3.2.1	traceRouteMaxConcurrentRequests	8
3.2.2	traceRouteCtlTable	8
3.2.3	traceRouteResultsTable	9

3.2.4	traceRouteProbeHistoryTable	9
3.2.5	traceRouteHopsTable	10
3.3	Lookup MIB	10
3.3.1	lookupMaxConcurrentRequests and lookupPurgeTime	10
3.3.2	lookupCtlTable	10
3.3.3	lookupResultsTable	11
4.0	Definitions	12
4.1	DISMAN-PING-MIB	12
4.2	DISMAN-TRACEROUTE-MIB	36
4.3	DISMAN-NSLOOKUP-MIB	63
5.0	Security Considerations	73
6.0	Intellectual Property	74
7.0	Acknowledgments	74
8.0	References	74
9.0	Author's Address	76
10.0	Full Copyright Statement	77

1.0 Introduction

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119, reference [13].

This document is a product of the Distributed Management (DISMAN) Working Group. Its purpose is to define standards-based MIB modules for performing specific remote operations. The remote operations defined by this document consist of the ping, traceroute and lookup functions.

Ping and traceroute are two very useful functions for managing networks. Ping is typically used to determine if a path exists between two hosts while traceroute shows an actual path. Ping is usually implemented using the Internet Control Message Protocol (ICMP) "ECHO" facility. It is also possible to implement a ping capability using alternate methods, some of which are:

- o Using the UDP echo port (7), if supported.

This is defined by RFC 862 [2].

- o Timing an SNMP query.
- o Timing a TCP connect attempt.

In general, almost any request/response flow can be used to generate a round-trip time. Often many of the non-ICMP ECHO facility methods stand a better chance of yielding a good response (not timing out for

example) since some routers don't honor Echo Requests (timeout situation) or they are handled at lower priority, hence possibly giving false indications of round trip times.

It must be noted that almost any of the various methods used for generating a round-trip time can be considered a form of system attack when used excessively. Sending a system requests too often can negatively effect its performance. Attempting to connect to what is supposed to be an unused port can be very unpredictable. There are tools that attempt to connect to a range of TCP ports to test that any receiving server can handle erroneous connection attempts.

It also is important to the management application using a remote ping capability to know which method is being used. Different methods will yield different response times since the protocol and resulting processing will be different. It is RECOMMENDED that the ping capability defined within this memo be implemented using the ICMP Echo Facility.

Traceroute is usually implemented by transmitting a series of probe packets with increasing time-to-live values. A probe packet is a UDP datagram encapsulated into an IP packet. Each hop in a path to the target (destination) host rejects the probe packet (probe's TTL too small) until its time-to-live value becomes large enough for the probe to be forwarded. Each hop in a traceroute path returns an ICMP message that is used to discover the hop and to calculate a round trip time. Some systems use ICMP probes (ICMP Echo request packets) instead of UDP ones to implement traceroute. In both cases traceroute relies on the probes being rejected via an ICMP message to discover the hops taken along a path to the final destination. Both probe types, UDP and ICMP, are encapsulated into an IP packet and thus have a TTL field that can be used to cause a path rejection.

Implementations of the remote traceroute capability as defined within this memo SHOULD be done using UDP packets to a (hopefully) unused port. ICMP probes (ICMP Echo Request packets) SHOULD NOT be used. Many PC implementations of traceroute use the ICMP probe method, which they should not, since this implementation method has been known to have a high probability of failure. Intermediate hops become invisible when a router either refuses to send an ICMP TTL expired message in response to an incoming ICMP packet or simply tosses ICMP echo requests altogether.

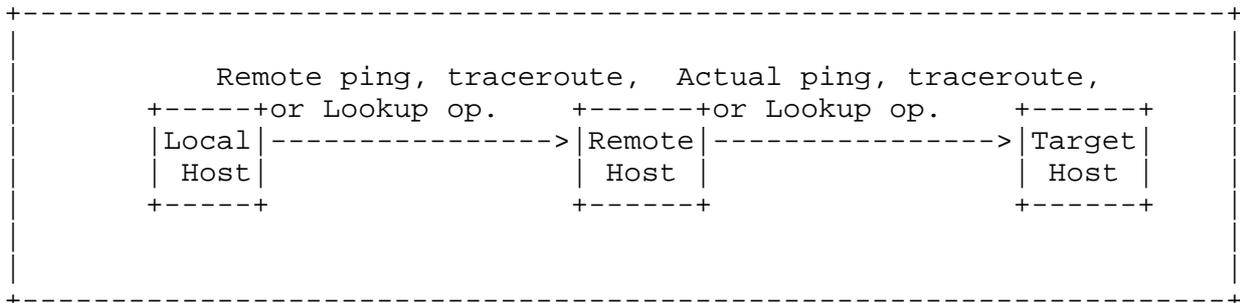
The behavior of some routers not to return a TTL expired message in response to an ICMP Echo request is due in part to the following text extracted from RFC 792 [20]:

"The ICMP messages typically report errors in the processing of datagrams. To avoid the infinite regress of messages about messages etc., no ICMP messages are sent about ICMP messages."

Both ping and traceroute yield round-trip times measured in milliseconds. These times can be used as a rough approximation for network transit time.

The Lookup operation enables the equivalent of either a `gethostbyname()` or a `gethostbyaddr()` call being performed at a remote host. The Lookup `gethostbyname()` capability can be used to determine the symbolic name of a hop in a traceroute path.

Consider the following diagram:



A local host is the host from which the remote ping, traceroute, or Lookup operation is initiated using an SNMP request. The remote host is a host where the MIBs defined by this memo are implemented that receives the remote operation via SNMP and performs the actual ping, traceroute, or lookup function.

2.0 The SNMP Network Management Framework

The SNMP Management Framework presently consists of five major components:

- o An overall architecture, described in RFC 2571 [7].
- o Mechanisms for describing and naming objects and events for the purpose of management. The first version of this Structure of Management Information (SMI) is called SMIV1 and described in STD 16, RFC 1155 [14], STD 16, RFC 1212 [15] and RFC 1215 [16]. The second version, called SMIV2, is described in STD 58, RFC 2578 [3], STD 58, RFC 2579 [4] and STD 58, RFC 2580 [5].

- o Message protocols for transferring management information. The first version of the SNMP message protocol is called SNMPv1 and described in STD 15, RFC 1157 [1]. A second version of the SNMP message protocol, which is not an Internet standards track protocol, is called SNMPv2c and described in RFC 1901 [17] and RFC 1906 [18]. The third version of the message protocol is called SNMPv3 and described in RFC 1906 [18], RFC 2572 [8] and RFC 2574 [10].
- o Protocol operations for accessing management information. The first set of protocol operations and associated PDU formats is described in STD 15, RFC 1157 [1]. A second set of protocol operations and associated PDU formats is described in RFC 1905 [6].
- o A set of fundamental applications described in RFC 2573 [9] and the view-based access control mechanism described in RFC 2575 [11].

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

This memo specifies MIB modules that are compliant to the SMIV2. A MIB conforming to the SMIV1 can be produced through the appropriate translations. The resulting translated MIB must be semantically equivalent, except where objects or events are omitted because no translation is possible (use of Counter64). Some machine readable information in SMIV2 will be converted into textual descriptions in SMIV1 during the translation process. However, this loss of machine readable information is not considered to change the semantics of the MIB.

3.0 Structure of the MIBs

This document defines three MIB modules:

- o DISMAN-PING-MIB
Defines a ping MIB.
- o DISMAN-TRACEROUTE-MIB
Defines a traceroute MIB.

- o DISMAN-NSLOOKUP-MIB

Provides access to the resolver `gethostbyname()` and `gethostbyaddr()` functions at a remote host.

The ping and traceroute MIBs are structured to allow creation of ping or traceroute tests that can be set up to periodically issue a series of operations and generate NOTIFICATIONS to report on test results. Many network administrators have in the past written UNIX shell scripts or command batch files to operate in fashion similar to the functionality provided by the ping and traceroute MIBs defined within this memo. The intent of this document is to acknowledge the importance of these functions and to provide a standards-based solution.

3.1 Ping MIB

The DISMAN-PING-MIB consists of the following components:

- o pingMaxConcurrentRequests
- o pingCtlTable
- o pingResultsTable
- o pingProbeHistoryTable

3.1.1 pingMaxConcurrentRequests

The object `pingMaxConcurrentRequests` enables control of the maximum number of concurrent active requests that an agent implementation supports. It is permissible for an agent either to limit the maximum upper range allowed for this object or to implement this object as read-only with an implementation limit expressed as its value.

3.1.2 pingCtlTable

A remote ping test is started by setting `pingCtlAdminStatus` to `enabled(1)`. The corresponding `pingCtlEntry` MUST have been created and its `pingCtlRowStatus` set to `active(1)` prior to starting the test. A single SNMP PDU can be used to create and start a remote ping test. Within the PDU, `pingCtlTargetAddress` should be set to the target host's address (`pingCtlTargetAddressType` will default to `ipv4(1)`), `pingCtlAdminStatus` to `enabled(1)`, and `pingCtlRowStatus` to `createAndGo(4)`.

The first index element, `pingCtlOwnerIndex`, is of type `SnmpAdminString`, a textual convention that allows for use of the SNMPv3 View-Based Access Control Model (RFC 2575 [11], VACM) and allows a management application to identify its entries. The second index, `pingCtlTestName` (also an `SnmpAdminString`), enables the same management application to have multiple requests outstanding.

Using the maximum value for the parameters defined within a `pingEntry` can result in a single remote ping test taking at most 15 minutes (`pingCtlTimeout` times `pingCtlProbeCount`) plus whatever time it takes to send the ping request and receive its response over the network from the target host. Use of the defaults for `pingCtlTimeout` and `pingCtlProbeCount` yields a maximum of 3 seconds to perform a "normal" ping test.

A management application can delete an active remote ping request by setting the corresponding `pingCtlRowStatus` object to `destroy(6)`.

The contents of the `pingCtlTable` is preserved across reIPLs (Initial Program Loads) of its agent according to the values of each of the `pingCtlStorageType` objects.

3.1.3 `pingResultsTable`

An entry in the `pingResultsTable` is created for a corresponding `pingCtlEntry` once the test defined by this entry is started.

3.1.4 `pingProbeHistoryTable`

The results of past ping probes can be stored in this table on a per `pingCtlEntry` basis. This table is initially indexed by `pingCtlOwnerIndex` and `pingCtlTestName` in order for the results of a probe to relate to the `pingCtlEntry` that caused it. The maximum number of entries stored in this table per `pingCtlEntry` is determined by the value of `pingCtlMaxRows`.

An implementation of this MIB will remove the oldest entry in the `pingProbeHistoryTable` to allow the addition of a new entry once the number of rows in the `pingProbeHistoryTable` reaches the value specified by `pingCtlMaxRows`. An implementation **MUST** start assigning `pingProbeHistoryIndex` values at 1 and wrap after exceeding the maximum possible value as defined by the limit of this object ('ffffffff'h).

3.2 Traceroute MIB

The DISMAN-TRACEROUTE-MIB consists of the following components:

- o traceRouteMaxConcurrentRequests
- o traceRouteCtlTable
- o traceRouteResultsTable
- o traceRouteProbeHistoryTable
- o traceRouteHopsTable

3.2.1 traceRouteMaxConcurrentRequests

The object traceRouteMaxConcurrentRequests enables control of the maximum number of concurrent active requests that an agent implementation supports. It is permissible for an agent either to limit the maximum upper range allowed for this object or to implement this object as read-only with an implementation limit expressed as its value.

3.2.2 traceRouteCtlTable

A remote traceroute test is started by setting traceRouteCtlAdminStatus to enabled(1). The corresponding traceRouteCtlEntry MUST have been created and its traceRouteCtlRowStatus set to active(1) prior to starting the test. A single SNMP PDU can be used to create and start a remote traceroute test. Within the PDU, traceRouteCtlTargetAddress should be set to the target host's address (traceRouteCtlTargetAddressType will default to ipv4(1)), traceRouteCtlAdminStatus to enabled(1), and traceRouteCtlRowStatus to createAndGo(4).

The first index element, traceRouteCtlOwnerIndex, is of type SnmpAdminString, a textual convention that allows for use of the SNMPv3 View-Based Access Control Model (RFC 2575 [11], VACM) and allows a management application to identify its entries. The second index, traceRouteCtlTestName (also an SnmpAdminString), enables the same management application to have multiple requests outstanding.

Traceroute has a much longer theoretical maximum time for completion than ping. Basically 42 hours and 30 minutes (the product of traceRouteCtlTimeOut, traceRouteCtlProbesPerHop, and traceRouteCtlMaxTtl) plus some network transit time! Use of the defaults defined within an traceRouteCtlEntry yields a maximum of 4 minutes and 30 seconds for a default traceroute operation. Clearly

42 plus hours is too long to wait for a traceroute operation to complete.

The maximum TTL value in effect for traceroute determines how long the traceroute function will keep increasing the TTL value in the probe it transmits hoping to reach the target host. The function ends whenever the maximum TTL is exceeded or the target host is reached. The object `traceRouteCtlMaxFailures` was created in order to impose a throttle for how long traceroute continues to increase the TTL field in a probe without receiving any kind of response (timeouts). It is RECOMMENDED that agent implementations impose a time limit for how long it allows a traceroute operation to take relative to how the function is implemented. For example, an implementation that can't process multiple traceroute operations at the same time SHOULD impose a shorter maximum allowed time period.

A management application can delete an active remote traceroute request by setting the corresponding `traceRouteCtlRowStatus` object to `destroy(6)`.

The contents of the `traceRouteCtlTable` is preserved across reIPLs (Initial Program Loads) of its agent according to the values of each of the `traceRouteCtlStorageType` objects.

3.2.3 `traceRouteResultsTable`

An entry in the `traceRouteResultsTable` is created upon determining the results of a specific traceroute operation. Entries in this table relate back to the `traceRouteCtlEntry` that caused the corresponding traceroute operation to occur. The objects `traceRouteResultsCurHopCount` and `traceRouteResultsCurProbeCount` can be examined to determine how far the current remote traceroute operation has reached.

3.2.4 `traceRouteProbeHistoryTable`

The results of past traceroute probes can be stored in this table on a per `traceRouteCtlEntry` basis. This table is initially indexed by `traceRouteCtlOwnerIndex` and `traceRouteCtlTestName` in order for the results of a probe to relate to the `traceRouteCtlEntry` that caused it. The number of entries stored in this table per `traceRouteCtlEntry` is determined by the value of `traceRouteCtlMaxRows`.

An implementation of this MIB will remove the oldest entry in the `traceRouteProbeHistoryTable` to allow the addition of a new entry once the number of rows in the `traceRouteProbeHistoryTable` reaches the value of `traceRouteCtlMaxRows`. An implementation MUST start

assigning `traceRouteProbeHistoryIndex` values at 1 and wrap after exceeding the maximum possible value as defined by the limit of this object ('ffffffff'h).

3.2.5 `traceRouteHopsTable`

The current traceroute path can be stored in this table on a per `traceRouteCtlEntry` basis. This table is initially indexed by `traceRouteCtlOwnerIndex` and `traceRouteCtlTestName` in order for a traceroute path to relate to the `traceRouteCtlEntry` that caused it. A third index, `traceRouteHopsHopIndex`, enables keeping one `traceRouteHopsEntry` per traceroute hop. Creation of `traceRouteHopsTable` entries is enabled by setting the corresponding `traceRouteCtlCreateHopsEntries` object to `true(1)`.

3.3 Lookup MIB

The DISMAN-NSLOOKUP-MIB consists of the following components:

- o `lookupMaxConcurrentRequests`, and `lookupPurgeTime`
- o `lookupCtlTable`
- o `lookupResultsTable`

3.3.1 `lookupMaxConcurrentRequests` and `lookupPurgeTime`

The object `lookupMaxConcurrentRequests` enables control of the maximum number of concurrent active requests that an agent implementation is structured to support. It is permissible for an agent either to limit the maximum upper range allowed for this object or to implement this object as read-only with an implementation limit expressed as its value.

The object `lookupPurgeTime` provides a method for entries in the `lookupCtlTable` and `lookupResultsTable` to be automatically deleted after the corresponding operation completes.

3.3.2 `lookupCtlTable`

A remote lookup operation is initiated by performing an SNMP SET request on `lookupCtlRowStatus`. A single SNMP PDU can be used to create and start a remote lookup operation. Within the PDU, `lookupCtlTargetAddress` should be set to the entity to be resolved (`lookupCtlTargetAddressType` will default to `ipv4(1)`) and `lookupCtlRowStatus` to `createAndGo(4)`. The object `lookupCtlOperStatus`

can be examined to determine the state of an lookup operation. A management application can delete an active remote lookup request by setting the corresponding lookupCtlRowStatus object to destroy(6).

An lookupCtlEntry is initially indexed by lookupCtlOwnerIndex, which is of type SnmpAdminString, a textual convention that allows for use of the SNMPv3 View-Based Access Control Model (RFC 2575 [11], VACM) and also allows for a management application to identify its entries. The lookupCtlOwnerIndex portion of the index is then followed by lookupCtlOperationName. The lookupCtlOperationName index enables the same lookupCtlOwnerIndex entity to have multiple outstanding requests.

The value of lookupCtlTargetAddressType determines which lookup function to perform. Specification of dns(16) as the value of this index implies that the gethostbyname function should be performed to determine the numeric addresses associated with a symbolic name via lookupResultsTable entries. Use of a value of either ipv4(1) or ipv6(2) implies that the gethostbyaddr function should be performed to determine the symbolic name(s) associated with a numeric address at a remote host.

3.3.3 lookupResultsTable

The lookupResultsTable is used to store the results of lookup operations. The lookupResultsTable is initially indexed by the same index elements that the lookupCtlTable contains (lookupCtlOwnerIndex and lookupCtlOperationName) but has a third index element, lookupResultsIndex (Unsigned32 textual convention), in order to associate multiple results with the same lookupCtlEntry.

Both the gethostbyname and gethostbyaddr functions typically return a pointer to a hostent structure after being called. The hostent structure is defined as:

```
struct hostent {
    char *h_name;          /* official host name      */
    char *h_aliases[];    /* list of other aliases   */
    int h_addrtype;       /* host address type       */
    int h_length;         /* length of host address  */
    char **h_addr_list;   /* list of address for host */
};
```

The hostent structure is listed here in order to address the fact that a remote host can be multi-homed and can have multiple symbolic (DNS) names. It is not intended to imply that implementations of the DISMAN-LOOKUP-MIB are limited to systems where the hostent structure is supported.

The `gethostbyaddr` function is called with a host address as its parameter and is used primarily to determine a symbolic name to associate with the host address. Entries in the `lookupResultsTable` MUST be made for each host name returned. The official host name MUST be assigned a `lookupResultsIndex` of 1.

The `gethostbyname` function is called with a symbolic host name and is used primarily to retrieve a host address. Normally, the first `h_addr_list` host address is considered to be the primary address and as such is associated with the symbolic name passed on the call.

Entries MUST be stored in the `lookupResultsTable` in the order that they are retrieved. Values assigned to `lookupResultsIndex` MUST start at 1 and increase in order.

An implementation SHOULD NOT retain SNMP-created entries in the `lookupTable` across reIPLs (Initial Program Loads) of its agent, since management applications need to see consistent behavior with respect to the persistence of the table entries that they create.

4.0 Definitions

4.1 DISMAN-PING-MIB

```
DISMAN-PING-MIB DEFINITIONS ::= BEGIN
```

```
IMPORTS
```

```
  MODULE-IDENTITY, OBJECT-TYPE, Integer32,
  Unsigned32, mib-2,
  NOTIFICATION-TYPE, OBJECT-IDENTITY
    FROM SNMPv2-SMI                -- RFC2578
  TEXTUAL-CONVENTION, RowStatus,
  StorageType, DateAndTime, TruthValue
    FROM SNMPv2-TC                -- RFC2579
  MODULE-COMPLIANCE, OBJECT-GROUP,
  NOTIFICATION-GROUP
    FROM SNMPv2-CONF              -- RFC2580
  InterfaceIndexOrZero
    FROM IF-MIB                    -- RFC2863
  SnmpAdminString
    FROM SNMP-FRAMEWORK-MIB        -- RFC2571
  InetAddressType, InetAddress
    FROM INET-ADDRESS-MIB;        -- RFC2851
```

```
pingMIB MODULE-IDENTITY
```

```
  LAST-UPDATED "200009210000Z"      -- 21 September 2000
  ORGANIZATION "IETF Distributed Management Working Group"
  CONTACT-INFO
```

"Kenneth White

International Business Machines Corporation
Network Computing Software Division
Research Triangle Park, NC, USA

E-mail: wkenneth@us.ibm.com"

DESCRIPTION

"The Ping MIB (DISMAN-PING-MIB) provides the capability of controlling the use of the ping function at a remote host."

-- Revision history

REVISION "200009210000Z" -- 21 September 2000

DESCRIPTION

"Initial version, published as RFC 2925."

::= { mib-2 80 }

-- Textual Conventions

OperationResponseStatus ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"Used to report the result of an operation:

responseReceived(1) - Operation completes successfully.

unknown(2) - Operation failed due to unknown error.

internalError(3) - An implementation detected an error in its own processing that caused an operation to fail.

requestTimedOut(4) - Operation failed to receive a valid reply within the time limit imposed on it.

unknownDestinationAddress(5) - Invalid destination address.

noRouteToTarget(6) - Could not find a route to target.

interfaceInactiveToTarget(7) - The interface to be used in sending a probe is inactive without an alternate route existing.

arpFailure(8) - Unable to resolve a target address to a media specific address.

maxConcurrentLimitReached(9) - The maximum number of concurrent active operations would have been exceeded if the corresponding operation was allowed.

unableToResolveDnsName(10) - The DNS name specified was unable to be mapped to an IP address.

invalidHostAddress(11) - The IP address for a host

has been determined to be invalid. Examples of this are broadcast or multicast addresses."

```
SYNTAX INTEGER {
    responseReceived(1),
    unknown(2),
    internalError(3),
    requestTimedOut(4),
    unknownDestinationAddress(5),
    noRouteToTarget(6),
    interfaceInactiveToTarget(7),
    arpFailure(8),
    maxConcurrentLimitReached(9),
    unableToResolveDnsName(10),
    invalidHostAddress(11)
}
```

-- Top level structure of the MIB

```
pingNotifications          OBJECT IDENTIFIER ::= { pingMIB 0 }
pingObjects                OBJECT IDENTIFIER ::= { pingMIB 1 }
pingConformance           OBJECT IDENTIFIER ::= { pingMIB 2 }
```

-- The registration node (point) for ping implementation types

```
pingImplementationTypeDomains OBJECT IDENTIFIER ::= { pingMIB 3 }
```

```
pingIcmpEcho OBJECT-IDENTITY
```

```
STATUS current
```

```
DESCRIPTION
```

```
"Indicates that an implementation is using the Internet
Control Message Protocol (ICMP) 'ECHO' facility."
```

```
::= { pingImplementationTypeDomains 1 }
```

```
pingUdpEcho OBJECT-IDENTITY
```

```
STATUS current
```

```
DESCRIPTION
```

```
"Indicates that an implementation is using the UDP echo
port (7)."
```

```
REFERENCE
```

```
"RFC 862, 'Echo Protocol'."
```

```
::= { pingImplementationTypeDomains 2 }
```

```
pingSnmpQuery OBJECT-IDENTITY
```

```
STATUS current
```

```
DESCRIPTION
```

```
"Indicates that an implementation is an SNMP query to
calculate a round trip time."
```

```
::= { pingImplementationTypeDomains 3 }
```

```
pingTcpConnectionAttempt OBJECT-IDENTITY
```

```
STATUS current
```

```
DESCRIPTION
```

```
"Indicates that an implementation is attempting to
connect to a TCP port in order to calculate a round
trip time."
```

```
::= { pingImplementationTypeDomains 4 }
```

```
-- Simple Object Definitions
```

```
pingMaxConcurrentRequests OBJECT-TYPE
```

```
SYNTAX Unsigned32
```

```
UNITS "requests"
```

```
MAX-ACCESS read-write
```

```
STATUS current
```

```
DESCRIPTION
```

```
"The maximum number of concurrent active ping requests
that are allowed within an agent implementation. A value
of 0 for this object implies that there is no limit for
the number of concurrent active requests in effect."
```

```
DEFVAL { 10 }
```

```
::= { pingObjects 1 }
```

```
-- Ping Control Table
```

```
pingCtlTable OBJECT-TYPE
```

```
SYNTAX SEQUENCE OF PingCtlEntry
```

```
MAX-ACCESS not-accessible
```

```
STATUS current
```

```
DESCRIPTION
```

```
"Defines the ping Control Table for providing, via SNMP,
the capability of performing ping operations at
a remote host. The results of these operations are
stored in the pingResultsTable and the
pingProbeHistoryTable."
```

```
::= { pingObjects 2 }
```

```
pingCtlEntry OBJECT-TYPE
```

```
SYNTAX PingCtlEntry
```

```
MAX-ACCESS not-accessible
```

```
STATUS current
```

```
DESCRIPTION
```

```
"Defines an entry in the pingCtlTable. The first index
element, pingCtlOwnerIndex, is of type SnmpAdminString,
a textual convention that allows for use of the SNMPv3
```

View-Based Access Control Model (RFC 2575 [11], VACM) and allows an management application to identify its entries. The second index, pingCtlTestName (also an SnmpAdminString), enables the same management application to have multiple outstanding requests."

```
INDEX {
    pingCtlOwnerIndex,
    pingCtlTestName
}
 ::= { pingCtlTable 1 }
```

PingCtlEntry ::=

```
SEQUENCE {
    pingCtlOwnerIndex          SnmpAdminString,
    pingCtlTestName           SnmpAdminString,
    pingCtlTargetAddressType  InetAddressType,
    pingCtlTargetAddress      InetAddress,
    pingCtlDataSize           Unsigned32,
    pingCtlTimeOut            Unsigned32,
    pingCtlProbeCount         Unsigned32,
    pingCtlAdminStatus        INTEGER,
    pingCtlDataFill           OCTET STRING,
    pingCtlFrequency          Unsigned32,
    pingCtlMaxRows            Unsigned32,
    pingCtlStorageType        StorageType,
    pingCtlTrapGeneration     BITS,
    pingCtlTrapProbeFailureFilter Unsigned32,
    pingCtlTrapTestFailureFilter Unsigned32,
    pingCtlType               OBJECT IDENTIFIER,
    pingCtlDescr              SnmpAdminString,
    pingCtlSourceAddressType  InetAddressType,
    pingCtlSourceAddress      InetAddress,
    pingCtlIfIndex            InterfaceIndexOrZero,
    pingCtlByPassRouteTable   TruthValue,
    pingCtlDSField            Unsigned32,
    pingCtlRowStatus          RowStatus
}
```

pingCtlOwnerIndex OBJECT-TYPE

SYNTAX SnmpAdminString (SIZE(0..32))

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"To facilitate the provisioning of access control by a security administrator using the View-Based Access Control Model (RFC 2575, VACM) for tables in which multiple users may need to independently create or modify entries, the initial index is used as an 'owner

index'. Such an initial index has a syntax of SnmpAdminString, and can thus be trivially mapped to a securityName or groupName as defined in VACM, in accordance with a security policy.

When used in conjunction with such a security policy all entries in the table belonging to a particular user (or group) will have the same value for this initial index. For a given user's entries in a particular table, the object identifiers for the information in these entries will have the same subidentifiers (except for the 'column' subidentifier) up to the end of the encoded owner index. To configure VACM to permit access to this portion of the table, one would create vacmViewTreeFamilyTable entries with the value of vacmViewTreeFamilySubtree including the owner index portion, and vacmViewTreeFamilyMask 'wildcarding' the column subidentifier. More elaborate configurations are possible."

```
::= { pingCtlEntry 1 }
```

pingCtlTestName OBJECT-TYPE

SYNTAX SnmpAdminString (SIZE(0..32))

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"The name of the ping test. This is locally unique, within the scope of an pingCtlOwnerIndex."

```
::= { pingCtlEntry 2 }
```

pingCtlTargetAddressType OBJECT-TYPE

SYNTAX InetAddressType

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"Specifies the type of host address to be used at a remote host for performing a ping operation."

DEFVAL { unknown }

```
::= { pingCtlEntry 3 }
```

pingCtlTargetAddress OBJECT-TYPE

SYNTAX InetAddress

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"Specifies the host address to be used at a remote host for performing a ping operation. The host address type is determined by the object value of corresponding pingCtlTargetAddressType."

A value for this object MUST be set prior to transitioning its corresponding pingCtlEntry to active(1) via pingCtlRowStatus."

```
DEFVAL { 'H }  
 ::= { pingCtlEntry 4 }
```

pingCtlDataSize OBJECT-TYPE

```
SYNTAX      Unsigned32 (0..65507)
```

```
UNITS       "octets"
```

```
MAX-ACCESS  read-create
```

```
STATUS      current
```

DESCRIPTION

"Specifies the size of the data portion to be transmitted in a ping operation in octets. A ping request is usually an ICMP message encoded into an IP packet. An IP packet has a maximum size of 65535 octets. Subtracting the size of the ICMP or UDP header (both 8 octets) and the size of the IP header (20 octets) yields a maximum size of 65507 octets."

```
DEFVAL { 0 }  
 ::= { pingCtlEntry 5 }
```

pingCtlTimeOut OBJECT-TYPE

```
SYNTAX      Unsigned32 (1..60)
```

```
UNITS       "seconds"
```

```
MAX-ACCESS  read-create
```

```
STATUS      current
```

DESCRIPTION

"Specifies the time-out value, in seconds, for a remote ping operation."

```
DEFVAL { 3 }  
 ::= { pingCtlEntry 6 }
```

pingCtlProbeCount OBJECT-TYPE

```
SYNTAX      Unsigned32 (1..15)
```

```
UNITS       "probes"
```

```
MAX-ACCESS  read-create
```

```
STATUS      current
```

DESCRIPTION

"Specifies the number of times to perform a ping operation at a remote host."

```
DEFVAL { 1 }  
 ::= { pingCtlEntry 7 }
```

pingCtlAdminStatus OBJECT-TYPE

```
SYNTAX      INTEGER {  
                enabled(1), -- test should be started
```

```

                disabled(2) -- test should be stopped
            }
MAX-ACCESS    read-create
STATUS        current
DESCRIPTION   "Reflects the desired state that a pingCtlEntry should be
in:

                enabled(1) - Attempt to activate the test as defined by
                this pingCtlEntry.
                disabled(2) - Deactivate the test as defined by this
                pingCtlEntry.

Refer to the corresponding pingResultsOperStatus to
determine the operational state of the test defined by
this entry."
DEFVAL { disabled }
 ::= { pingCtlEntry 8 }

```

```

pingCtlDataFill OBJECT-TYPE
SYNTAX        OCTET STRING (SIZE(0..1024))
MAX-ACCESS    read-create
STATUS        current
DESCRIPTION   "The content of this object is used together with the
corresponding pingCtlDataSize value to determine how to
fill the data portion of a probe packet. The option of
selecting a data fill pattern can be useful when links
are compressed or have data pattern sensitivities. The
contents of pingCtlDataFill should be repeated in a ping
packet when the size of the data portion of the ping
packet is greater than the size of pingCtlDataFill."
DEFVAL { '00'H }
 ::= { pingCtlEntry 9 }

```

```

pingCtlFrequency OBJECT-TYPE
SYNTAX        Unsigned32
UNITS         "seconds"
MAX-ACCESS    read-create
STATUS        current
DESCRIPTION   "The number of seconds to wait before repeating a ping test
as defined by the value of the various objects in the
corresponding row.

A single ping test consists of a series of ping probes.
The number of probes is determined by the value of the
corresponding pingCtlProbeCount object. After a single

```

test completes the number of seconds as defined by the value of pingCtlFrequency MUST elapse before the next ping test is started.

A value of 0 for this object implies that the test as defined by the corresponding entry will not be repeated."

```
DEFVAL { 0 }
 ::= { pingCtlEntry 10 }
```

pingCtlMaxRows OBJECT-TYPE

```
SYNTAX      Unsigned32
UNITS       "rows"
MAX-ACCESS  read-create
STATUS      current
```

DESCRIPTION

"The maximum number of entries allowed in the pingProbeHistoryTable. An implementation of this MIB will remove the oldest entry in the pingProbeHistoryTable to allow the addition of a new entry once the number of rows in the pingProbeHistoryTable reaches this value.

Old entries are not removed when a new test is started. Entries are added to the pingProbeHistoryTable until pingCtlMaxRows is reached before entries begin to be removed.

A value of 0 for this object disables creation of pingProbeHistoryTable entries."

```
DEFVAL      { 50 }
 ::= { pingCtlEntry 11 }
```

pingCtlStorageType OBJECT-TYPE

```
SYNTAX      StorageType
MAX-ACCESS  read-create
STATUS      current
```

DESCRIPTION

"The storage type for this conceptual row. Conceptual rows having the value 'permanent' need not allow write-access to any columnar objects in the row."

```
DEFVAL { nonVolatile }
 ::= { pingCtlEntry 12 }
```

pingCtlTrapGeneration OBJECT-TYPE

```
SYNTAX      BITS {
                probeFailure(0),
                testFailure(1),
```

```

        testCompletion(2)
    }
MAX-ACCESS read-create
STATUS current
DESCRIPTION
    "The value of this object determines when and if
    to generate a notification for this entry:

    probeFailure(0) - Generate a pingProbeFailed
    notification subject to the value of
    pingCtlTrapProbeFailureFilter. The object
    pingCtlTrapProbeFailureFilter can be used
    to specify the number of successive probe failures
    that are required before a pingProbeFailed
    notification can be generated.
    testFailure(1) - Generate a pingTestFailed
    notification. In this instance the object
    pingCtlTrapTestFailureFilter can be used to
    determine the number of probe failures that
    signal when a test fails.
    testCompletion(2) - Generate a pingTestCompleted
    notification.
```

The value of this object defaults to zero, indicating that none of the above options have been selected."

```
::= { pingCtlEntry 13 }
```

pingCtlTrapProbeFailureFilter OBJECT-TYPE

SYNTAX Unsigned32 (0..15)

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"The value of this object is used to determine when to generate a pingProbeFailed NOTIFICATION.

Setting pingCtlTrapGeneration to probeFailure(0) implies that a pingProbeFailed NOTIFICATION is generated only when the number of successive probe failures as indicated by the value of pingCtlTrapPrbfailureFilter fail within a given ping test."

DEFVAL { 1 }

```
::= { pingCtlEntry 14 }
```

pingCtlTrapTestFailureFilter OBJECT-TYPE

SYNTAX Unsigned32 (0..15)

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"The value of this object is used to determine when to generate a pingTestFailed NOTIFICATION.

Setting pingCtlTrapGeneration to testFailure(1) implies that a pingTestFailed NOTIFICATION is generated only when the number of ping failures within a test exceed the value of pingCtlTrapTestFailureFilter."

DEFVAL { 1 }

::= { pingCtlEntry 15 }

pingCtlType OBJECT-TYPE

SYNTAX OBJECT IDENTIFIER

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"The value of this object is used to either report or select the implementation method to be used for calculating a ping response time. The value of this object MAY be selected from pingImplementationTypeDomains.

Additional implementation types SHOULD be allocated as required by implementers of the DISMAN-PING-MIB under their enterprise specific registration point and not beneath pingImplementationTypeDomains."

DEFVAL { pingIcmpEcho }

::= { pingCtlEntry 16 }

pingCtlDescr OBJECT-TYPE

SYNTAX SnmpAdminString

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"The purpose of this object is to provide a descriptive name of the remote ping test."

DEFVAL { '00'H }

::= { pingCtlEntry 17 }

pingCtlSourceAddressType OBJECT-TYPE

SYNTAX InetAddressType

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"Specifies the type of the source address, pingCtlSourceAddress, to be used at a remote host when performing a ping operation."

DEFVAL { ipv4 }

```
::= { pingCtlEntry 18 }
```

```
pingCtlSourceAddress OBJECT-TYPE
```

```
SYNTAX      InetAddress
```

```
MAX-ACCESS  read-create
```

```
STATUS      current
```

```
DESCRIPTION
```

"Use the specified IP address (which must be given in numeric form, not as a hostname) as the source address in outgoing probe packets. On hosts with more than one IP address, this option can be used to force the source address to be something other than the primary IP address of the interface the probe packet is sent on. If the IP address is not one of this machine's interface addresses, an error is returned and nothing is sent. A zero length octet string value for this object disables source address specification.

The address type (InetAddressType) that relates to this object is specified by the corresponding value of pingCtlSourceAddressType."

```
DEFVAL { ''H }
```

```
::= { pingCtlEntry 19 }
```

```
pingCtlIfIndex OBJECT-TYPE
```

```
SYNTAX      InterfaceIndexOrZero
```

```
MAX-ACCESS  read-create
```

```
STATUS      current
```

```
DESCRIPTION
```

"Setting this object to an interface's ifIndex prior to starting a remote ping operation directs the ping probes to be transmitted over the specified interface. A value of zero for this object means that this option is not enabled."

```
DEFVAL { 0 }
```

```
::= { pingCtlEntry 20 }
```

```
pingCtlByPassRouteTable OBJECT-TYPE
```

```
SYNTAX      TruthValue
```

```
MAX-ACCESS  read-create
```

```
STATUS      current
```

```
DESCRIPTION
```

"The purpose of this object is to optionally enable bypassing the route table. If enabled, the remote host will bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly-attached network, an

error is returned. This option can be used to perform the ping operation to a local host through an interface that has no route defined (e.g., after the interface was dropped by routed)."

DEFVAL { false }
 ::= { pingCtlEntry 21 }

pingCtlDSField OBJECT-TYPE

SYNTAX Unsigned32 (0..255)

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"Specifies the value to store in the Differentiated Services (DS) Field in the IP packet used to encapsulate the ping probe. The DS Field is defined as the Type of Service (TOS) octet in a IPv4 header or as the Traffic Class octet in a IPv6 header.

The value of this object must be a decimal integer in the range from 0 to 255. This option can be used to determine what effect an explicit DS Field setting has on a ping response. Not all values are legal or meaningful. A value of 0 means that the function represented by this option is not supported. DS Field usage is often not supported by IP implementations and not all values are supported. Refer to RFC 2474 for guidance on usage of this field."

REFERENCE

"Refer to RFC 2474 for the definition of the Differentiated Services Field and to RFC 1812 Section 5.3.2 for Type of Service (TOS)."

DEFVAL { 0 }
 ::= { pingCtlEntry 22 }

pingCtlRowStatus OBJECT-TYPE

SYNTAX RowStatus

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"This object allows entries to be created and deleted in the pingCtlTable. Deletion of an entry in this table results in all corresponding (same pingCtlOwnerIndex and pingCtlTestName index values) pingResultsTable and pingProbeHistoryTable entries being deleted.

A value MUST be specified for pingCtlTargetAddress prior to a transition to active(1) state being

accepted.

Activation of a remote ping operation is controlled via pingCtlAdminStatus and not by changing this object's value to active(1).

Transitions in and out of active(1) state are not allowed while an entry's pingResultsOperStatus is active(1) with the exception that deletion of an entry in this table by setting its RowStatus object to destroy(6) will stop an active ping operation.

The operational state of a ping operation can be determined by examination of its pingResultsOperStatus object."

REFERENCE

"See definition of RowStatus in RFC 2579, 'Textual Conventions for SMIV2.'"

::= { pingCtlEntry 23 }

-- Ping Results Table

pingResultsTable OBJECT-TYPE

SYNTAX SEQUENCE OF PingResultsEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"Defines the Ping Results Table for providing the capability of performing ping operations at a remote host. The results of these operations are stored in the pingResultsTable and the pingPastProbeTable.

An entry is added to the pingResultsTable when an pingCtlEntry is started by successful transition of its pingCtlAdminStatus object to enabled(1). An entry is removed from the pingResultsTable when its corresponding pingCtlEntry is deleted."

::= { pingObjects 3 }

pingResultsEntry OBJECT-TYPE

SYNTAX PingResultsEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"Defines an entry in the pingResultsTable. The pingResultsTable has the same indexing as the pingCtlTable in order for a pingResultsEntry to

correspond to the pingCtlEntry that caused it to be created."

```

INDEX {
    pingCtlOwnerIndex,
    pingCtlTestName
}
 ::= { pingResultsTable 1 }

PingResultsEntry ::=
SEQUENCE {
    pingResultsOperStatus          INTEGER,
    pingResultsIpTargetAddressType InetAddressType,
    pingResultsIpTargetAddress     InetAddress,
    pingResultsMinRtt              Unsigned32,
    pingResultsMaxRtt              Unsigned32,
    pingResultsAverageRtt          Unsigned32,
    pingResultsProbeResponses      Unsigned32,
    pingResultsSentProbes          Unsigned32,
    pingResultsRttSumOfSquares     Unsigned32,
    pingResultsLastGoodProbe       DateAndTime
}

pingResultsOperStatus OBJECT-TYPE
SYNTAX          INTEGER {
                enabled(1), -- test is in progress
                disabled(2) -- test has stopped
                }
MAX-ACCESS     read-only
STATUS         current
DESCRIPTION   "Reflects the operational state of a pingCtlEntry:
                enabled(1) - Test is active.
                disabled(2) - Test has stopped."
 ::= { pingResultsEntry 1 }

pingResultsIpTargetAddressType OBJECT-TYPE
SYNTAX          InetAddressType
MAX-ACCESS     read-only
STATUS         current
DESCRIPTION   "This objects indicates the type of address stored
                in the corresponding pingResultsIpTargetAddress
                object."
DEFVAL { unknown }
 ::= { pingResultsEntry 2 }

pingResultsIpTargetAddress OBJECT-TYPE
SYNTAX          InetAddress

```

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This objects reports the IP address associated with a pingCtlTargetAddress value when the destination address is specified as a DNS name. The value of this object should be a zero length octet string when a DNS name is not specified or when a specified DNS name fails to resolve."

DEFVAL { ''H }

::= { pingResultsEntry 3 }

pingResultsMinRtt OBJECT-TYPE

SYNTAX Unsigned32

UNITS "milliseconds"

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The minimum ping round-trip-time (RTT) received. A value of 0 for this object implies that no RTT has been received."

::= { pingResultsEntry 4 }

pingResultsMaxRtt OBJECT-TYPE

SYNTAX Unsigned32

UNITS "milliseconds"

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The maximum ping round-trip-time (RTT) received. A value of 0 for this object implies that no RTT has been received."

::= { pingResultsEntry 5 }

pingResultsAverageRtt OBJECT-TYPE

SYNTAX Unsigned32

UNITS "milliseconds"

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The current average ping round-trip-time (RTT)."

::= { pingResultsEntry 6 }

pingResultsProbeResponses OBJECT-TYPE

SYNTAX Unsigned32

UNITS "responses"

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"Number of responses received for the corresponding

pingCtlEntry and pingResultsEntry. The value of this object MUST be reported as 0 when no probe responses have been received."

::= { pingResultsEntry 7 }

pingResultsSentProbes OBJECT-TYPE

SYNTAX Unsigned32

UNITS "probes"

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The value of this object reflects the number of probes sent for the corresponding pingCtlEntry and pingResultsEntry. The value of this object MUST be reported as 0 when no probes have been sent."

::= { pingResultsEntry 8 }

pingResultsRttSumOfSquares OBJECT-TYPE

SYNTAX Unsigned32

UNITS "milliseconds"

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object contains the sum of the squares for all ping responses received. Its purpose is to enable standard deviation calculation. The value of this object MUST be reported as 0 when no ping responses have been received."

::= { pingResultsEntry 9 }

pingResultsLastGoodProbe OBJECT-TYPE

SYNTAX DateAndTime

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"Date and time when the last response was received for a probe."

::= { pingResultsEntry 10 }

-- Ping Probe History Table

pingProbeHistoryTable OBJECT-TYPE

SYNTAX SEQUENCE OF PingProbeHistoryEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"Defines a table for storing the results of a ping operation. Entries in this table are limited by

the value of the corresponding pingCtlMaxRows object.

An entry in this table is created when the result of a ping probe is determined. The initial 2 instance identifier index values identify the pingCtlEntry that a probe result (pingProbeHistoryEntry) belongs to. An entry is removed from this table when its corresponding pingCtlEntry is deleted.

An implementation of this MIB will remove the oldest entry in the pingProbeHistoryTable to allow the addition of an new entry once the number of rows in the pingProbeHistoryTable reaches the value specified by pingCtlMaxRows."

```
::= { pingObjects 4 }
```

pingProbeHistoryEntry OBJECT-TYPE

SYNTAX PingProbeHistoryEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"Defines an entry in the pingProbeHistoryTable. The first two index elements identify the pingCtlEntry that a pingProbeHistoryEntry belongs to. The third index element selects a single probe result."

```
INDEX {
    pingCtlOwnerIndex,
    pingCtlTestName,
    pingProbeHistoryIndex
}
```

```
::= { pingProbeHistoryTable 1 }
```

PingProbeHistoryEntry ::=

```
SEQUENCE {
    pingProbeHistoryIndex      Unsigned32,
    pingProbeHistoryResponse   Unsigned32,
    pingProbeHistoryStatus     OperationResponseStatus,
    pingProbeHistoryLastRC     Integer32,
    pingProbeHistoryTime       DateAndTime
}
```

pingProbeHistoryIndex OBJECT-TYPE

SYNTAX Unsigned32 (1..'ffffffff'h)

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"An entry in this table is created when the result of a ping probe is determined. The initial 2 instance identifier index values identify the pingCtlEntry that a probe result (pingProbeHistoryEntry) belongs to.

An implementation MUST start assigning pingProbeHistoryIndex values at 1 and wrap after exceeding the maximum possible value as defined by the limit of this object ('ffffffff'h')."

```
::= { pingProbeHistoryEntry 1 }
```

pingProbeHistoryResponse OBJECT-TYPE

SYNTAX Unsigned32

UNITS "milliseconds"

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The amount of time measured in milliseconds from when a probe was sent to when its response was received or when it timed out. The value of this object is reported as 0 when it is not possible to transmit a probe."

```
::= { pingProbeHistoryEntry 2 }
```

pingProbeHistoryStatus OBJECT-TYPE

SYNTAX OperationResponseStatus

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The result of a particular probe done by a remote host."

```
::= { pingProbeHistoryEntry 3 }
```

pingProbeHistoryLastRC OBJECT-TYPE

SYNTAX Integer32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The last implementation method specific reply code received. If the ICMP Echo capability is being used then a successful probe ends when an ICMP response is received that contains the code ICMP_ECHOREPLY(0). The ICMP responses are defined normally in the ip_icmp include file."

```
::= { pingProbeHistoryEntry 4 }
```

pingProbeHistoryTime OBJECT-TYPE

SYNTAX DateAndTime

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"Timestamp for when this probe result was determined."

::= { pingProbeHistoryEntry 5 }

-- Notification Definition section

pingProbeFailed NOTIFICATION-TYPE

```
OBJECTS {
    pingCtlTargetAddressType,
    pingCtlTargetAddress,
    pingResultsOperStatus,
    pingResultsIpTargetAddressType,
    pingResultsIpTargetAddress,
    pingResultsMinRtt,
    pingResultsMaxRtt,
    pingResultsAverageRtt,
    pingResultsProbeResponses,
    pingResultsSentProbes,
    pingResultsRttSumOfSquares,
    pingResultsLastGoodProbe
}
```

STATUS current

DESCRIPTION

"Generated when a probe failure is detected when the corresponding pingCtlTrapGeneration object is set to probeFailure(0) subject to the value of pingCtlTrapProbeFailureFilter. The object pingCtlTrapProbeFailureFilter can be used to specify the number of successive probe failures that are required before this notification can be generated."

::= { pingNotifications 1 }

pingTestFailed NOTIFICATION-TYPE

```
OBJECTS {
    pingCtlTargetAddressType,
    pingCtlTargetAddress,
    pingResultsOperStatus,
    pingResultsIpTargetAddressType,
    pingResultsIpTargetAddress,
    pingResultsMinRtt,
    pingResultsMaxRtt,
    pingResultsAverageRtt,
    pingResultsProbeResponses,
    pingResultsSentProbes,
    pingResultsRttSumOfSquares,
    pingResultsLastGoodProbe
}
```

```

STATUS    current
DESCRIPTION
    "Generated when a ping test is determined to have failed
    when the corresponding pingCtlTrapGeneration object is
    set to testFailure(1).  In this instance
    pingCtlTrapTestFailureFilter should specify the number of
    probes in a test required to have failed in order to
    consider the test as failed."
 ::= { pingNotifications 2 }

pingTestCompleted NOTIFICATION-TYPE
OBJECTS {
    pingCtlTargetAddressType,
    pingCtlTargetAddress,
    pingResultsOperStatus,
    pingResultsIpTargetAddressType,
    pingResultsIpTargetAddress,
    pingResultsMinRtt,
    pingResultsMaxRtt,
    pingResultsAverageRtt,
    pingResultsProbeResponses,
    pingResultsSentProbes,
    pingResultsRttSumOfSquares,
    pingResultsLastGoodProbe
}
STATUS    current
DESCRIPTION
    "Generated at the completion of a ping test when the
    corresponding pingCtlTrapGeneration object is set to
    testCompletion(4)."
```

```

 ::= { pingNotifications 3 }

-- Conformance information
-- Compliance statements

pingCompliances OBJECT IDENTIFIER ::= { pingConformance 1 }
pingGroups      OBJECT IDENTIFIER ::= { pingConformance 2 }

-- Compliance statements

pingCompliance MODULE-COMPLIANCE
STATUS    current
DESCRIPTION
    "The compliance statement for the DISMAN-PING-MIB."
MODULE   -- this module
MANDATORY-GROUPS {
    pingGroup,
    pingNotificationsGroup

```

```
    }
GROUP pingTimeStampGroup
DESCRIPTION
    "This group is mandatory for implementations that have
    access to a system clock and are capable of setting
    the values for DateAndTime objects. It is RECOMMENDED
    that when this group is not supported that the values
    for the objects in this group be reported as
    '0000000000000000'H."

OBJECT pingMaxConcurrentRequests
MIN-ACCESS read-only
DESCRIPTION
    "The agent is not required to support set
    operations to this object."

OBJECT pingCtlStorageType
MIN-ACCESS read-only
DESCRIPTION
    "Write access is not required. It is also allowed
    for implementations to support only the volatile
    StorageType enumeration."

OBJECT pingCtlType
MIN-ACCESS read-only
DESCRIPTION
    "Write access is not required. In addition, the only
    value that MUST be supported by an implementation is
    pingIcmpEcho."

OBJECT pingCtlByPassRouteTable
MIN-ACCESS read-only
DESCRIPTION
    "This object is not required by implementations that
    are not capable of its implementation. The function
    represented by this object is implementable if the
    setsockopt SOL_SOCKET SO_DONTROUTE option is
    supported."

OBJECT pingCtlSourceAddressType
SYNTAX InetAddressType { unknown(0), ipv4(1), ipv6(2) }
MIN-ACCESS read-only
DESCRIPTION
    "This object is not required by implementations that
    are not capable of binding the send socket with a
    source address. An implementation is only required to
    support IPv4 and IPv6 addresses."
```

```
OBJECT pingCtlSourceAddress
SYNTAX InetAddress (SIZE(0|4|16))
MIN-ACCESS read-only
DESCRIPTION
    "This object is not required by implementations that
    are not capable of binding the send socket with a
    source address. An implementation is only required to
    support IPv4 and globally unique IPv6 addresses."
```

```
OBJECT pingCtlIfIndex
MIN-ACCESS read-only
DESCRIPTION
    "Write access is not required. When write access is
    not supported return a 0 as the value of this object.
    A value of 0 means that the function represented by
    this option is not supported."
```

```
OBJECT pingCtlDSField
MIN-ACCESS read-only
DESCRIPTION
    "Write access is not required. When write access is
    not supported return a 0 as the value of this object.
    A value of 0 means that the function represented by
    this option is not supported."
```

```
OBJECT pingResultsIpTargetAddressType
SYNTAX InetAddressType { unknown(0), ipv4(1), ipv6(2) }
DESCRIPTION
    "An implementation is only required to
    support IPv4 and IPv6 addresses."
```

```
OBJECT pingResultsIpTargetAddress
SYNTAX InetAddress (SIZE(0|4|16))
DESCRIPTION
    "An implementation is only required to
    support IPv4 and globally unique IPv6 addresses."
```

```
::= { pingCompliances 1 }
```

```
-- MIB groupings
```

```
pingGroup OBJECT-GROUP
  OBJECTS {
    pingMaxConcurrentRequests,
    pingCtlTargetAddressType,
    pingCtlTargetAddress,
    pingCtlDataSize,
    pingCtlTimeOut,
```

```

    pingCtlProbeCount,
    pingCtlAdminStatus,
    pingCtlDataFill,
    pingCtlFrequency,
    pingCtlMaxRows,
    pingCtlStorageType,
    pingCtlTrapGeneration,
    pingCtlTrapProbeFailureFilter,
    pingCtlTrapTestFailureFilter,
    pingCtlType,
    pingCtlDescr,
    pingCtlByPassRouteTable,
    pingCtlSourceAddressType,
    pingCtlSourceAddress,
    pingCtlIfIndex,
    pingCtlDSField,
    pingCtlRowStatus,
    pingResultsOperStatus,
    pingResultsIpTargetAddressType,
    pingResultsIpTargetAddress,
    pingResultsMinRtt,
    pingResultsMaxRtt,
    pingResultsAverageRtt,
    pingResultsProbeResponses,
    pingResultsSentProbes,
    pingResultsRttSumOfSquares,
    pingProbeHistoryResponse,
    pingProbeHistoryStatus,
    pingProbeHistoryLastRC
  }
STATUS current
DESCRIPTION
  "The group of objects that comprise the remote ping
  capability."
 ::= { pingGroups 1 }

pingTimeStampGroup OBJECT-GROUP
OBJECTS {
    pingResultsLastGoodProbe,
    pingProbeHistoryTime
}
STATUS current
DESCRIPTION
  "The group of DateAndTime objects."
 ::= { pingGroups 2 }

pingNotificationsGroup NOTIFICATION-GROUP
NOTIFICATIONS {

```

```

        pingProbeFailed,
        pingTestFailed,
        pingTestCompleted
    }
STATUS          current
DESCRIPTION
    "The notification which are required to be supported by
    implementations of this MIB."
 ::= { pingGroups 3 }

```

END

4.2 DISMAN-TRACEROUTE-MIB

DISMAN-TRACEROUTE-MIB DEFINITIONS ::= BEGIN

IMPORTS

```

MODULE-IDENTITY, OBJECT-TYPE, Integer32,
Gauge32, Unsigned32, mib-2,
NOTIFICATION-TYPE,
OBJECT-IDENTITY
    FROM SNMPv2-SMI                -- RFC2578
RowStatus, StorageType,
TruthValue, DateAndTime
    FROM SNMPv2-TC                -- RFC2579
MODULE-COMPLIANCE, OBJECT-GROUP,
NOTIFICATION-GROUP
    FROM SNMPv2-CONF              -- RFC2580
SnmpAdminString
    FROM SNMP-FRAMEWORK-MIB        -- RFC2571
InterfaceIndexOrZero
    FROM IF-MIB
InetAddressType, InetAddress
    FROM INET-ADDRESS-MIB         -- RFC2851
OperationResponseStatus
    FROM DISMAN-PING-MIB;         -- RFC2925

```

traceRouteMIB MODULE-IDENTITY

```

    LAST-UPDATED "200009210000Z"    -- 21 September 2000

```

```

    ORGANIZATION "IETF Distributed Management Working Group"

```

```

    CONTACT-INFO

```

```

        "Kenneth White

```

```

        International Business Machines Corporation
        Network Computing Software Division
        Research Triangle Park, NC, USA

```

E-mail: wkenneth@us.ibm.com"

DESCRIPTION

"The Traceroute MIB (DISMAN-TRACEROUTE-MIB) provides access to the traceroute capability at a remote host."

-- Revision history

REVISION "200009210000Z" -- 21 September 2000

DESCRIPTION

"Initial version, published as RFC 2925."

::= { mib-2 81 }

-- Top level structure of the MIB

```
traceRouteNotifications OBJECT IDENTIFIER ::= { traceRouteMIB 0 }
traceRouteObjects       OBJECT IDENTIFIER ::= { traceRouteMIB 1 }
traceRouteConformance  OBJECT IDENTIFIER ::= { traceRouteMIB 2 }
```

-- The registration node (point) for traceroute implementation types

```
traceRouteImplementationTypeDomains OBJECT IDENTIFIER
::= { traceRouteMIB 3 }
```

```
traceRouteUsingUdpProbes OBJECT-IDENTITY
```

```
STATUS current
```

DESCRIPTION

"Indicates that an implementation is using UDP probes to perform the traceroute operation."

```
::= { traceRouteImplementationTypeDomains 1 }
```

-- Simple Object Definitions

```
traceRouteMaxConcurrentRequests OBJECT-TYPE
```

```
SYNTAX Unsigned32
```

```
UNITS "requests"
```

```
MAX-ACCESS read-write
```

```
STATUS current
```

DESCRIPTION

"The maximum number of concurrent active traceroute requests that are allowed within an agent implementation. A value of 0 for this object implies that there is no limit for the number of concurrent active requests in effect."

```
DEFVAL { 10 }
```

```
::= { traceRouteObjects 1 }
```

-- Traceroute Control Table

traceRouteCtlTable OBJECT-TYPE

SYNTAX SEQUENCE OF TraceRouteCtlEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"Defines the Remote Operations Traceroute Control Table for providing the capability of invoking traceroute from a remote host. The results of traceroute operations can be stored in the traceRouteResultsTable, traceRouteProbeHistoryTable, and the traceRouteHopsTable."

::= { traceRouteObjects 2 }

traceRouteCtlEntry OBJECT-TYPE

SYNTAX TraceRouteCtlEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"Defines an entry in the traceRouteCtlTable. The first index element, traceRouteCtlOwnerIndex, is of type SnmpAdminString, a textual convention that allows for use of the SNMPv3 View-Based Access Control Model (RFC 2575 [11], VACM) and allows an management application to identify its entries. The second index, traceRouteCtlTestName (also an SnmpAdminString), enables the same management application to have multiple requests outstanding."

INDEX {
 traceRouteCtlOwnerIndex,
 traceRouteCtlTestName
 }

::= { traceRouteCtlTable 1 }

TraceRouteCtlEntry ::=

SEQUENCE {

traceRouteCtlOwnerIndex	SnmpAdminString,
traceRouteCtlTestName	SnmpAdminString,
traceRouteCtlTargetAddressType	InetAddressType,
traceRouteCtlTargetAddress	InetAddress,
traceRouteCtlByPassRouteTable	TruthValue,
traceRouteCtlDataSize	Unsigned32,
traceRouteCtlTimeOut	Unsigned32,
traceRouteCtlProbesPerHop	Unsigned32,
traceRouteCtlPort	Unsigned32,
traceRouteCtlMaxTtl	Unsigned32,
traceRouteCtlDSField	Unsigned32,
traceRouteCtlSourceAddressType	InetAddressType,

```

traceRouteCtlSourceAddress      InetAddress,
traceRouteCtlIfIndex            InterfaceIndexOrZero,
traceRouteCtlMiscOptions        SnmpAdminString,
traceRouteCtlMaxFailures        Unsigned32,
traceRouteCtlDontFragment       TruthValue,
traceRouteCtlInitialTtl         Unsigned32,
traceRouteCtlFrequency          Unsigned32,
traceRouteCtlStorageType        StorageType,
traceRouteCtlAdminStatus        INTEGER,
traceRouteCtlMaxRows            Unsigned32,
traceRouteCtlTrapGeneration     BITS,
traceRouteCtlDescr              SnmpAdminString,
traceRouteCtlCreateHopsEntries  TruthValue,
traceRouteCtlType                OBJECT IDENTIFIER,
traceRouteCtlRowStatus           RowStatus
}

```

```

traceRouteCtlOwnerIndex OBJECT-TYPE
SYNTAX      SnmpAdminString (SIZE(0..32))
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION

```

"To facilitate the provisioning of access control by a security administrator using the View-Based Access Control Model (RFC 2575, VACM) for tables in which multiple users may need to independently create or modify entries, the initial index is used as an 'owner index'. Such an initial index has a syntax of SnmpAdminString, and can thus be trivially mapped to a securityName or groupName as defined in VACM, in accordance with a security policy.

When used in conjunction with such a security policy all entries in the table belonging to a particular user (or group) will have the same value for this initial index. For a given user's entries in a particular table, the object identifiers for the information in these entries will have the same subidentifiers (except for the 'column' subidentifier) up to the end of the encoded owner index. To configure VACM to permit access to this portion of the table, one would create vacmViewTreeFamilyTable entries with the value of vacmViewTreeFamilySubtree including the owner index portion, and vacmViewTreeFamilyMask 'wildcarding' the column subidentifier. More elaborate configurations are possible."

```
 ::= { traceRouteCtlEntry 1 }
```

```
traceRouteCtlTestName OBJECT-TYPE
    SYNTAX      SnmpAdminString (SIZE(0..32))
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The name of a traceroute test.  This is locally unique,
        within the scope of an traceRouteCtlOwnerIndex."
    ::= { traceRouteCtlEntry 2 }
```

```
traceRouteCtlTargetAddressType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "Specifies the type of host address to be used on the
        traceroute request at the remote host."
    DEFVAL { ipv4 }
    ::= { traceRouteCtlEntry 3 }
```

```
traceRouteCtlTargetAddress OBJECT-TYPE
    SYNTAX      InetAddress
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "Specifies the host address used on the
        traceroute request at the remote host.  The
        host address type can be determined by the
        examining the value of the corresponding
        traceRouteCtlTargetAddressType index element.

        A value for this object MUST be set prior to
        transitioning its corresponding traceRouteCtlEntry to
        active(1) via traceRouteCtlRowStatus."
    ::= { traceRouteCtlEntry 4 }
```

```
traceRouteCtlByPassRouteTable OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "The purpose of this object is to optionally enable
        bypassing the route table.  If enabled, the remote
        host will bypass the normal routing tables and send
        directly to a host on an attached network.  If the
        host is not on a directly-attached network, an
        error is returned.  This option can be used to perform
        the traceroute operation to a local host through an
        interface that has no route defined (e.g., after the
```

```
        interface was dropped by routed)."
    DEFVAL { false }
    ::= { traceRouteCtlEntry 5 }

traceRouteCtlDataSize OBJECT-TYPE
    SYNTAX      Unsigned32 (0..65507)
    UNITS       "octets"
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "Specifies the size of the data portion of a traceroute
        request in octets.  A traceroute request is essentially
        transmitted by encoding a UDP datagram into a
        IP packet.  So subtracting the size of a UDP header
        (8 octets) and the size of a IP header (20 octets)
        yields a maximum of 65507 octets."
    DEFVAL { 0 }
    ::= { traceRouteCtlEntry 6 }

traceRouteCtlTimeOut OBJECT-TYPE
    SYNTAX      Unsigned32 (1..60)
    UNITS       "seconds"
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "Specifies the time-out value, in seconds, for
        a traceroute request."
    DEFVAL { 3 }
    ::= { traceRouteCtlEntry 7 }

traceRouteCtlProbesPerHop OBJECT-TYPE
    SYNTAX      Unsigned32 (1..10)
    UNITS       "probes"
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "Specifies the number of times to reissue a traceroute
        request with the same time-to-live (TTL) value."
    DEFVAL { 3 }
    ::= { traceRouteCtlEntry 8 }

traceRouteCtlPort OBJECT-TYPE
    SYNTAX      Unsigned32 (1..65535)
    UNITS       "UDP Port"
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "Specifies the UDP port to send the traceroute
```

request to. Need to specify a port that is not in use at the destination (target) host. The default value for this object is the IANA assigned port, 33434, for the traceroute function."

DEFVAL { 33434 }
 ::= { traceRouteCtlEntry 9 }

traceRouteCtlMaxTtl OBJECT-TYPE

SYNTAX Unsigned32 (1..255)
UNITS "time-to-live value"
MAX-ACCESS read-create
STATUS current
DESCRIPTION

"Specifies the maximum time-to-live value."

DEFVAL { 30 }
 ::= { traceRouteCtlEntry 10 }

traceRouteCtlDSField OBJECT-TYPE

SYNTAX Unsigned32 (0..255)
MAX-ACCESS read-create
STATUS current
DESCRIPTION

"Specifies the value to store in the Differentiated Services (DS) Field in the IP packet used to encapsulate the traceroute probe. The DS Field is defined as the Type of Service (TOS) octet in a IPv4 header or as the Traffic Class octet in a IPv6 header.

The value of this object must be a decimal integer in the range from 0 to 255. This option can be used to determine what effect an explicit DS Field setting has on a traceroute response. Not all values are legal or meaningful. DS Field usage is often not supported by IP implementations. A value of 0 means that the function represented by this option is not supported. Useful TOS octet values are probably '16' (low delay) and '8' (high throughput)."

REFERENCE

"Refer to RFC 2474 for the definition of the Differentiated Services Field and to RFC 1812 Section 5.3.2 for Type of Service (TOS)."

DEFVAL { 0 }
 ::= { traceRouteCtlEntry 11 }

traceRouteCtlSourceAddressType OBJECT-TYPE

SYNTAX InetAddressType
MAX-ACCESS read-create
STATUS current

DESCRIPTION

"Specifies the type of the source address, traceRouteCtlSourceAddress, to be used at a remote host when performing a traceroute operation."

DEFVAL { unknown }

::= { traceRouteCtlEntry 12 }

traceRouteCtlSourceAddress OBJECT-TYPE

SYNTAX InetAddress

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"Use the specified IP address (which must be given as an IP number, not a hostname) as the source address in outgoing probe packets. On hosts with more than one IP address, this option can be used to force the source address to be something other than the primary IP address of the interface the probe packet is sent on. If the IP address is not one of this machine's interface addresses, an error is returned and nothing is sent. A zero length octet string value for this object disables source address specification.

The address type (InetAddressType) that relates to this object is specified by the corresponding value of traceRouteCtlSourceAddressType."

DEFVAL { ''H }

::= { traceRouteCtlEntry 13 }

traceRouteCtlIfIndex OBJECT-TYPE

SYNTAX InterfaceIndexOrZero

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"Setting this object to an interface's ifIndex prior to starting a remote traceroute operation directs the traceroute probes to be transmitted over the specified interface. A value of zero for this object implies that this option is not enabled."

DEFVAL { 0 }

::= { traceRouteCtlEntry 14 }

traceRouteCtlMiscOptions OBJECT-TYPE

SYNTAX SnmpAdminString

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"Enables an application to specify implementation dependent options."

DEFVAL { 'H }
 ::= { traceRouteCtlEntry 15 }

traceRouteCtlMaxFailures OBJECT-TYPE

SYNTAX Unsigned32 (0..255)

UNITS "timeouts"

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"The value of this object indicates the maximum number of consecutive timeouts allowed before terminating a remote traceroute request. A value of either 255 (maximum hop count/possible TTL value) or a 0 indicates that the function of terminating a remote traceroute request when a specific number of successive timeouts are detected is disabled."

DEFVAL { 5 }
 ::= { traceRouteCtlEntry 16 }

traceRouteCtlDontFragment OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"This object enables setting of the don't fragment flag (DF) in the IP header for a probe. Use of this object enables performing a manual PATH MTU test."

DEFVAL { false }
 ::= { traceRouteCtlEntry 17 }

traceRouteCtlInitialTtl OBJECT-TYPE

SYNTAX Unsigned32 (0..255)

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"The value of this object specifies the initial TTL value to use. This enables bypassing the initial (often well known) portion of a path."

DEFVAL { 1 }
 ::= { traceRouteCtlEntry 18 }

traceRouteCtlFrequency OBJECT-TYPE

SYNTAX Unsigned32

UNITS "seconds"

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"The number of seconds to wait before repeating a traceroute test as defined by the value of the various objects in the corresponding row.

The number of hops in a single traceroute test is determined by the value of the corresponding traceRouteCtlProbesPerHop object. After a single test completes the number of seconds as defined by the value of traceRouteCtlFrequency MUST elapse before the next traceroute test is started.

A value of 0 for this object implies that the test as defined by the corresponding entry will not be repeated."

```
DEFVAL { 0 }
 ::= { traceRouteCtlEntry 19 }
```

traceRouteCtlStorageType OBJECT-TYPE

```
SYNTAX      StorageType
MAX-ACCESS  read-create
STATUS      current
```

DESCRIPTION

"The storage type for this conceptual row. Conceptual rows having the value 'permanent' need not allow write-access to any columnar objects in the row."

```
DEFVAL { nonVolatile }
 ::= { traceRouteCtlEntry 20 }
```

traceRouteCtlAdminStatus OBJECT-TYPE

```
SYNTAX      INTEGER {
                enabled(1), -- operation should be started
                disabled(2) -- operation should be stopped
            }
```

```
MAX-ACCESS  read-create
STATUS      current
```

DESCRIPTION

"Reflects the desired state that an traceRouteCtlEntry should be in:

- enabled(1) - Attempt to activate the test as defined by this traceRouteCtlEntry.
- disabled(2) - Deactivate the test as defined by this traceRouteCtlEntry.

Refer to the corresponding traceRouteResultsOperStatus to determine the operational state of the test defined by this entry."

```
DEFVAL { disabled }
 ::= { traceRouteCtlEntry 21 }
```

traceRouteCtlDescr OBJECT-TYPE

SYNTAX SnmpAdminString

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"The purpose of this object is to provide a descriptive name of the remote traceroute test."

```
DEFVAL { '00'H }
```

```
 ::= { traceRouteCtlEntry 22 }
```

traceRouteCtlMaxRows OBJECT-TYPE

SYNTAX Unsigned32

UNITS "rows"

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"The maximum number of entries allowed in the traceRouteProbeHistoryTable. An implementation of this MIB will remove the oldest entry in the traceRouteProbeHistoryTable to allow the addition of a new entry once the number of rows in the traceRouteProbeHistoryTable reaches this value.

Old entries are not removed when a new test is started. Entries are added to the traceRouteProbeHistoryTable until traceRouteCtlMaxRows is reached before entries begin to be removed.

A value of 0 for this object disables creation of traceRouteProbeHistoryTable entries."

```
DEFVAL { 50 }
```

```
 ::= { traceRouteCtlEntry 23 }
```

traceRouteCtlTrapGeneration OBJECT-TYPE

```
SYNTAX BITS {
    pathChange(0),
    testFailure(1),
    testCompletion(2)
}
```

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"The value of this object determines when and if to generate a notification for this entry:

pathChange(0) - Generate a traceRoutePathChange notification when the current path varies from a previously determined path.

testFailure(1) - Generate a traceRouteTestFailed notification when the full path to a target can't be determined.

testCompletion(2) - Generate a traceRouteTestCompleted notification when the path to a target has been determined.

The value of this object defaults to zero, indicating that none of the above options have been selected."

::= { traceRouteCtlEntry 24 }

traceRouteCtlCreateHopsEntries OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"The current path for a traceroute test is kept in the traceRouteHopsTable on a per hop basis when the value of this object is true(1)."

DEFVAL { false }

::= { traceRouteCtlEntry 25 }

traceRouteCtlType OBJECT-TYPE

SYNTAX OBJECT IDENTIFIER

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"The value of this object is used either to report or select the implementation method to be used for performing a traceroute operation. The value of this object may be selected from traceRouteImplementationTypeDomains.

Additional implementation types should be allocated as required by implementers of the DISMAN-TRACEROUTE-MIB under their enterprise specific registration point and not beneath traceRouteImplementationTypeDomains."

DEFVAL { traceRouteUsingUdpProbes }

::= { traceRouteCtlEntry 26 }

traceRouteCtlRowStatus OBJECT-TYPE

SYNTAX RowStatus

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"This object allows entries to be created and deleted in the traceRouteCtlTable. Deletion of an entry in this table results in all corresponding (same traceRouteCtlOwnerIndex and traceRouteCtlTestName index values) traceRouteResultsTable, traceRouteProbeHistoryTable, and traceRouteHopsTable entries being deleted.

A value MUST be specified for traceRouteCtlTargetAddress prior to a transition to active(1) state being accepted.

Activation of a remote traceroute operation is controlled via traceRouteCtlAdminStatus and not by transitioning of this object's value to active(1).

Transitions in and out of active(1) state are not allowed while an entry's traceRouteResultsOperStatus is active(1) with the exception that deletion of an entry in this table by setting its RowStatus object to destroy(6) will stop an active traceroute operation.

The operational state of an traceroute operation can be determined by examination of the corresponding traceRouteResultsOperStatus object."

REFERENCE

"See definition of RowStatus in RFC 2579, 'Textual Conventions for SMIV2.'"

::= { traceRouteCtlEntry 27 }

-- Traceroute Results Table

traceRouteResultsTable OBJECT-TYPE

SYNTAX SEQUENCE OF TraceRouteResultsEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"Defines the Remote Operations Traceroute Results Table for keeping track of the status of a traceRouteCtlEntry.

An entry is added to the traceRouteResultsTable when an traceRouteCtlEntry is started by successful transition of its traceRouteCtlAdminStatus object to enabled(1). An entry is removed from the traceRouteResultsTable when its corresponding traceRouteCtlEntry is deleted."

::= { traceRouteObjects 3 }

```

traceRouteResultsEntry OBJECT-TYPE
    SYNTAX      TraceRouteResultsEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Defines an entry in the traceRouteResultsTable.  The
        traceRouteResultsTable has the same indexing as the
        traceRouteCtlTable in order for a traceRouteResultsEntry
        to correspond to the traceRouteCtlEntry that caused it to
        be created."
    INDEX {
        traceRouteCtlOwnerIndex,
        traceRouteCtlTestName
    }
    ::= { traceRouteResultsTable 1 }

TraceRouteResultsEntry ::=
    SEQUENCE {
        traceRouteResultsOperStatus      INTEGER,
        traceRouteResultsCurHopCount     Gauge32,
        traceRouteResultsCurProbeCount   Gauge32,
        traceRouteResultsIpTgtAddrType    InetAddressType,
        traceRouteResultsIpTgtAddr        InetAddress,
        traceRouteResultsTestAttempts     Unsigned32,
        traceRouteResultsTestSuccesses    Unsigned32,
        traceRouteResultsLastGoodPath     DateAndTime
    }

traceRouteResultsOperStatus OBJECT-TYPE
    SYNTAX      INTEGER {
        enabled(1), -- test is in progress
        disabled(2) -- test has stopped
    }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Reflects the operational state of an traceRouteCtlEntry:

        enabled(1) - Test is active.
        disabled(2) - Test has stopped."
    ::= { traceRouteResultsEntry 1 }

traceRouteResultsCurHopCount OBJECT-TYPE
    SYNTAX      Gauge32
    UNITS       "hops"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION

```

"Reflects the current TTL value (range from 1 to 255) for a remote traceroute operation. Maximum TTL value is determined by traceRouteCtlMaxTtl."

::= { traceRouteResultsEntry 2 }

traceRouteResultsCurProbeCount OBJECT-TYPE

SYNTAX Gauge32
UNITS "probes"
MAX-ACCESS read-only
STATUS current

DESCRIPTION

"Reflects the current probe count (1..10) for a remote traceroute operation. The maximum probe count is determined by traceRouteCtlProbesPerHop."

::= { traceRouteResultsEntry 3 }

traceRouteResultsIpTgtAddrType OBJECT-TYPE

SYNTAX InetAddressType
MAX-ACCESS read-only
STATUS current

DESCRIPTION

"This objects indicates the type of address stored in the corresponding traceRouteResultsIpTgtAddr object."

::= { traceRouteResultsEntry 4 }

traceRouteResultsIpTgtAddr OBJECT-TYPE

SYNTAX InetAddress
MAX-ACCESS read-only
STATUS current

DESCRIPTION

"This objects reports the IP address associated with a traceRouteCtlTargetAddress value when the destination address is specified as a DNS name. The value of this object should be a zero length octet string when a DNS name is not specified or when a specified DNS name fails to resolve."

::= { traceRouteResultsEntry 5 }

traceRouteResultsTestAttempts OBJECT-TYPE

SYNTAX Unsigned32
UNITS "tests"
MAX-ACCESS read-only
STATUS current

DESCRIPTION

"The current number of attempts to determine a path

to a target. The value of this object MUST be started at 0."

::= { traceRouteResultsEntry 6 }

traceRouteResultsTestSuccesses OBJECT-TYPE

SYNTAX Unsigned32

UNITS "tests"

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The current number of attempts to determine a path to a target that have succeeded. The value of this object MUST be reported as 0 when no attempts have succeeded."

::= { traceRouteResultsEntry 7 }

traceRouteResultsLastGoodPath OBJECT-TYPE

SYNTAX DateAndTime

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The date and time when the last complete path was determined."

::= { traceRouteResultsEntry 8 }

-- Trace Route Probe History Table

traceRouteProbeHistoryTable OBJECT-TYPE

SYNTAX SEQUENCE OF TraceRouteProbeHistoryEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"Defines the Remote Operations Traceroute Results Table for storing the results of a traceroute operation.

An implementation of this MIB will remove the oldest entry in the traceRouteProbeHistoryTable to allow the addition of a new entry once the number of rows in the traceRouteProbeHistoryTable reaches the value specified by traceRouteCtlMaxRows."

::= { traceRouteObjects 4 }

traceRouteProbeHistoryEntry OBJECT-TYPE

SYNTAX TraceRouteProbeHistoryEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"Defines a table for storing the results of a traceroute

operation. Entries in this table are limited by the value of the corresponding traceRouteCtlMaxRows object.

The first two index elements identify the traceRouteCtlEntry that a traceRouteProbeHistoryEntry belongs to. The third index element selects a single traceroute operation result. The fourth and fifth indexes select the hop and the probe for a particular traceroute operation."

```
INDEX {
    traceRouteCtlOwnerIndex,
    traceRouteCtlTestName,
    traceRouteProbeHistoryIndex,
    traceRouteProbeHistoryHopIndex,
    traceRouteProbeHistoryProbeIndex
}
 ::= { traceRouteProbeHistoryTable 1 }
```

```
TraceRouteProbeHistoryEntry ::=
SEQUENCE {
    traceRouteProbeHistoryIndex          Unsigned32,
    traceRouteProbeHistoryHopIndex       Unsigned32,
    traceRouteProbeHistoryProbeIndex     Unsigned32,
    traceRouteProbeHistoryHAddrType      InetAddressType,
    traceRouteProbeHistoryHAddr           InetAddress,
    traceRouteProbeHistoryResponse       Unsigned32,
    traceRouteProbeHistoryStatus          OperationResponseStatus,
    traceRouteProbeHistoryLastRC         Integer32,
    traceRouteProbeHistoryTime           DateAndTime
}
```

```
traceRouteProbeHistoryIndex OBJECT-TYPE
SYNTAX      Unsigned32 (1..'ffffffff'h)
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
```

"An entry in this table is created when the result of a traceroute probe is determined. The initial 2 instance identifier index values identify the traceRouteCtlEntry that a probe result (traceRouteProbeHistoryEntry) belongs to. An entry is removed from this table when its corresponding traceRouteCtlEntry is deleted.

An implementation MUST start assigning traceRouteProbeHistoryIndex values at 1 and wrap after exceeding the maximum possible value as defined by the limit of this object ('ffffffff'h')."

```
::= { traceRouteProbeHistoryEntry 1 }
```

```
traceRouteProbeHistoryHopIndex OBJECT-TYPE
```

```
SYNTAX      Unsigned32 (1..255)
```

```
MAX-ACCESS  not-accessible
```

```
STATUS      current
```

```
DESCRIPTION
```

```
"Indicates which hop in a traceroute path that the probe's
results are for.  The value of this object is initially
determined by the value of traceRouteCtlInitialTtl."
```

```
::= { traceRouteProbeHistoryEntry 2 }
```

```
traceRouteProbeHistoryProbeIndex OBJECT-TYPE
```

```
SYNTAX      Unsigned32 (1..10)
```

```
MAX-ACCESS  not-accessible
```

```
STATUS      current
```

```
DESCRIPTION
```

```
"Indicates the index of a probe for a particular
hop in a traceroute path.  The number of probes per
hop is determined by the value of the corresponding
traceRouteCtlProbesPerHop object."
```

```
::= { traceRouteProbeHistoryEntry 3 }
```

```
traceRouteProbeHistoryHAddrType OBJECT-TYPE
```

```
SYNTAX      InetAddressType
```

```
MAX-ACCESS  read-only
```

```
STATUS      current
```

```
DESCRIPTION
```

```
"This objects indicates the type of address stored
in the corresponding traceRouteProbeHistoryHAddr
object."
```

```
::= { traceRouteProbeHistoryEntry 4 }
```

```
traceRouteProbeHistoryHAddr OBJECT-TYPE
```

```
SYNTAX      InetAddress
```

```
MAX-ACCESS  read-only
```

```
STATUS      current
```

```
DESCRIPTION
```

```
"The address of a hop in a traceroute path.  This object
is not allowed to be a DNS name.  The value of the
corresponding object, traceRouteProbeHistoryHAddrType,
indicates this object's IP address type."
```

```
::= { traceRouteProbeHistoryEntry 5 }
```

```
traceRouteProbeHistoryResponse OBJECT-TYPE
```

```
SYNTAX      Unsigned32
```

```
UNITS       "milliseconds"
```

```
MAX-ACCESS  read-only
```

STATUS current

DESCRIPTION

"The amount of time measured in milliseconds from when a probe was sent to when its response was received or when it timed out. The value of this object is reported as 0 when it is not possible to transmit a probe."

::= { traceRouteProbeHistoryEntry 6 }

traceRouteProbeHistoryStatus OBJECT-TYPE

SYNTAX OperationResponseStatus

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The result of a traceroute operation made by a remote host for a particular probe."

::= { traceRouteProbeHistoryEntry 7 }

traceRouteProbeHistoryLastRC OBJECT-TYPE

SYNTAX Integer32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The last implementation method specific reply code received.

Traceroute is usually implemented by transmitting a series of probe packets with increasing time-to-live values. A probe packet is a UDP datagram encapsulated into an IP packet. Each hop in a path to the target (destination) host rejects the probe packets (probe's TTL too small, ICMP reply) until either the maximum TTL is exceeded or the target host is received."

::= { traceRouteProbeHistoryEntry 8 }

traceRouteProbeHistoryTime OBJECT-TYPE

SYNTAX DateAndTime

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"Timestamp for when this probe results were determined."

::= { traceRouteProbeHistoryEntry 9 }

-- Traceroute Hop Results Table

traceRouteHopsTable OBJECT-TYPE

SYNTAX SEQUENCE OF TraceRouteHopsEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"Defines the Remote Operations Traceroute Hop Table for keeping track of the results of traceroute tests on a per hop basis."

```
::= { traceRouteObjects 5 }
```

```
traceRouteHopsEntry OBJECT-TYPE
```

```
SYNTAX      TraceRouteHopsEntry
```

```
MAX-ACCESS not-accessible
```

```
STATUS      current
```

```
DESCRIPTION
```

"Defines an entry in the traceRouteHopsTable.

The first two index elements identify the traceRouteCtlEntry that a traceRouteHopsEntry belongs to. The third index element, traceRouteHopsHopIndex, selects a hop in a traceroute path."

```
INDEX {
    traceRouteCtlOwnerIndex,
    traceRouteCtlTestName,
    traceRouteHopsHopIndex
}
```

```
::= { traceRouteHopsTable 1 }
```

```
TraceRouteHopsEntry ::=
```

```
SEQUENCE {
    traceRouteHopsHopIndex          Unsigned32,
    traceRouteHopsIpTgtAddressType InetAddressType,
    traceRouteHopsIpTgtAddress     InetAddress,
    traceRouteHopsMinRtt           Unsigned32,
    traceRouteHopsMaxRtt           Unsigned32,
    traceRouteHopsAverageRtt       Unsigned32,
    traceRouteHopsRttSumOfSquares  Unsigned32,
    traceRouteHopsSentProbes       Unsigned32,
    traceRouteHopsProbeResponses   Unsigned32,
    traceRouteHopsLastGoodProbe    DateAndTime
}
```

```
traceRouteHopsHopIndex OBJECT-TYPE
```

```
SYNTAX      Unsigned32
```

```
MAX-ACCESS not-accessible
```

```
STATUS      current
```

```
DESCRIPTION
```

"Specifies the hop index for a traceroute hop. Values for this object with respect to the same traceRouteCtlOwnerIndex and traceRouteCtlTestName MUST start at 1 and increase monotonically.

The traceRouteHopsTable keeps the current traceroute path per traceRouteCtlEntry if enabled by setting the corresponding traceRouteCtlCreateHopsEntries to true(1).

All hops (traceRouteHopsTable entries) in a traceroute path MUST be updated at the same time when a traceroute operation completes. Care needs to be applied when either a path changes or can't be determined. The initial portion of the path, up to the first hop change, MUST retain the same traceRouteHopsHopIndex values. The remaining portion of the path SHOULD be assigned new traceRouteHopsHopIndex values."

```
::= { traceRouteHopsEntry 1 }
```

```
traceRouteHopsIpTgtAddressType OBJECT-TYPE
```

```
SYNTAX      InetAddressType
```

```
MAX-ACCESS  read-only
```

```
STATUS      current
```

```
DESCRIPTION
```

```
"This objects indicates the type of address stored in the corresponding traceRouteHopsIpTargetAddress object."
```

```
::= { traceRouteHopsEntry 2 }
```

```
traceRouteHopsIpTgtAddress OBJECT-TYPE
```

```
SYNTAX      InetAddress
```

```
MAX-ACCESS  read-only
```

```
STATUS      current
```

```
DESCRIPTION
```

```
"This object reports the IP address associated with the hop. A value for this object should be reported as a numeric IP address and not as a DNS name."
```

```
::= { traceRouteHopsEntry 3 }
```

```
traceRouteHopsMinRtt OBJECT-TYPE
```

```
SYNTAX      Unsigned32
```

```
MAX-ACCESS  read-only
```

```
STATUS      current
```

```
DESCRIPTION
```

```
"The minimum traceroute round-trip-time (RTT) received for this hop. A value of 0 for this object implies that no RTT has been received."
```

```
::= { traceRouteHopsEntry 4 }
```

```
traceRouteHopsMaxRtt OBJECT-TYPE
```

```
SYNTAX      Unsigned32
```

```
MAX-ACCESS  read-only
```

```
STATUS      current
DESCRIPTION
    "The maximum traceroute round-trip-time (RTT) received for
    this hop.  A value of 0 for this object implies that no
    RTT has been received."
 ::= { traceRouteHopsEntry 5 }

traceRouteHopsAverageRtt OBJECT-TYPE
SYNTAX      Unsigned32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The current average traceroute round-trip-time (RTT) for
    this hop."
 ::= { traceRouteHopsEntry 6 }

traceRouteHopsRttSumOfSquares OBJECT-TYPE
SYNTAX      Unsigned32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "This object contains the sum of all traceroute responses
    received for this hop.  Its purpose is to enable standard
    deviation calculation."
 ::= { traceRouteHopsEntry 7 }

traceRouteHopsSentProbes OBJECT-TYPE
SYNTAX      Unsigned32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The value of this object reflects the number of probes sent
    for this hop during this traceroute test.  The value of this
    object should start at 0."
 ::= { traceRouteHopsEntry 8 }

traceRouteHopsProbeResponses OBJECT-TYPE
SYNTAX      Unsigned32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "Number of responses received for this hop during this
    traceroute test.  This value of this object should start
    at 0."
 ::= { traceRouteHopsEntry 9 }

traceRouteHopsLastGoodProbe OBJECT-TYPE
SYNTAX      DateAndTime
```

```
MAX-ACCESS    read-only
STATUS        current
DESCRIPTION   "Date and time was the last response was received for a probe
              for this hop during this traceroute test."
 ::= { traceRouteHopsEntry 10 }

-- Notification Definition section

traceRoutePathChange NOTIFICATION-TYPE
  OBJECTS {
    traceRouteCtlTargetAddressType,
    traceRouteCtlTargetAddress,
    traceRouteResultsIpTgtAddrType,
    traceRouteResultsIpTgtAddr
  }
  STATUS      current
  DESCRIPTION "The path to a target has changed."
 ::= { traceRouteNotifications 1 }

traceRouteTestFailed NOTIFICATION-TYPE
  OBJECTS {
    traceRouteCtlTargetAddressType,
    traceRouteCtlTargetAddress,
    traceRouteResultsIpTgtAddrType,
    traceRouteResultsIpTgtAddr
  }
  STATUS      current
  DESCRIPTION "Could not determine the path to a target."
 ::= { traceRouteNotifications 2 }

traceRouteTestCompleted NOTIFICATION-TYPE
  OBJECTS {
    traceRouteCtlTargetAddressType,
    traceRouteCtlTargetAddress,
    traceRouteResultsIpTgtAddrType,
    traceRouteResultsIpTgtAddr
  }
  STATUS      current
  DESCRIPTION "The path to a target has just been determined."
 ::= { traceRouteNotifications 3 }

-- Conformance information
-- Compliance statements
```

```

traceRouteCompliances OBJECT IDENTIFIER ::= { traceRouteConformance 1 }
traceRouteGroups      OBJECT IDENTIFIER ::= { traceRouteConformance 2 }

-- Compliance statements

traceRouteCompliance MODULE-COMPLIANCE
  STATUS current
  DESCRIPTION
    "The compliance statement for the DISMAN-TRACEROUTE-MIB."
  MODULE -- this module
    MANDATORY-GROUPS {
      traceRouteGroup
    }
  GROUP traceRouteTimeStampGroup
  DESCRIPTION
    "This group is mandatory for implementations that have
    access to a system clock and are capable of setting
    the values for DateAndTime objects."

  GROUP traceRouteNotificationsGroup
  DESCRIPTION
    "This group defines a collection of optional
    notifications."

  GROUP traceRouteHopsTableGroup
  DESCRIPTION
    "This group lists the objects that make up a
    traceRouteHopsEntry. Support of the traceRouteHopsTable
    is optional."

  OBJECT traceRouteMaxConcurrentRequests
  MIN-ACCESS read-only
  DESCRIPTION
    "The agent is not required to support SET
    operations to this object."

  OBJECT traceRouteCtlByPassRouteTable
  MIN-ACCESS read-only
  DESCRIPTION
    "This object is not required by implementations that
    are not capable of its implementation. The function
    represented by this object is implementable if the
    setsockopt SOL_SOCKET SO_DONTROUTE option is
    supported."

  OBJECT traceRouteCtlSourceAddressType
  SYNTAX InetAddressType { unknown(0), ipv4(1), ipv6(2) }
  MIN-ACCESS read-only

```

DESCRIPTION

"This object is not required by implementations that are not capable of binding the send socket with a source address. An implementation is only required to support IPv4 and IPv6 addresses."

OBJECT traceRouteCtlSourceAddress
SYNTAX InetAddress (SIZE(0|4|16))
MIN-ACCESS read-only
DESCRIPTION

"This object is not required by implementations that are not capable of binding the send socket with a source address. An implementation is only required to support IPv4 and globally unique IPv6 addresses."

OBJECT traceRouteCtlIfIndex
MIN-ACCESS read-only
DESCRIPTION

"Write access is not required. When write access is not supported return a 0 as the value of this object. A value of 0 implies that the function represented by this option is not supported."

OBJECT traceRouteCtlMiscOptions
MIN-ACCESS read-only
DESCRIPTION

"Support of this object is optional. When not supporting do not allow write access and return a zero length octet string as the value of the object."

OBJECT traceRouteCtlStorageType
MIN-ACCESS read-only
DESCRIPTION

"Write access is not required. It is also allowed for implementations to support only the volatile StorageType enumeration."

OBJECT traceRouteCtlDSField
MIN-ACCESS read-only
DESCRIPTION

"Write access is not required. When write access is not supported return a 0 as the value of this object. A value of 0 implies that the function represented by this option is not supported."

OBJECT traceRouteCtlType
MIN-ACCESS read-only
DESCRIPTION

"Write access is not required. In addition, the only value that is RECOMMENDED to be supported by an implementation is traceRouteUsingUdpProbes."

OBJECT traceRouteResultsIpTgtAddrType
 SYNTAX InetAddressType { unknown(0), ipv4(1), ipv6(2) }
 DESCRIPTION
 "An implementation should only support IPv4 and globally unique IPv6 address values for this object."

OBJECT traceRouteResultsIpTgtAddr
 SYNTAX InetAddress (SIZE(0|4|16))
 DESCRIPTION
 "An implementation should only support IPv4 and globally unique IPv6 address values for this object."

OBJECT traceRouteProbeHistoryHAddrType
 SYNTAX InetAddressType { unknown(0), ipv4(1), ipv6(2) }
 DESCRIPTION
 "An implementation should only support IPv4 and globally unique IPv6 address values for this object."

OBJECT traceRouteProbeHistoryHAddr
 SYNTAX InetAddress (SIZE(0|4|16))
 DESCRIPTION
 "An implementation should only support IPv4 and globally unique IPv6 address values for this object."

OBJECT traceRouteHopsIpTgtAddressType
 SYNTAX InetAddressType { unknown(0), ipv4(1), ipv6(2) }
 DESCRIPTION
 "An implementation should only support IPv4 and globally unique IPv6 address values for this object."

OBJECT traceRouteHopsIpTgtAddress
 SYNTAX InetAddress (SIZE(0|4|16))
 DESCRIPTION
 "An implementation should only support IPv4 and globally unique IPv6 address values for this object."

::= { traceRouteCompliances 1 }

-- MIB groupings

traceRouteGroup OBJECT-GROUP
 OBJECTS {
 traceRouteMaxConcurrentRequests,
 traceRouteCtlTargetAddressType,
 traceRouteCtlTargetAddress,
 traceRouteCtlByPassRouteTable,

```

    traceRouteCtlDataSize,
    traceRouteCtlTimeOut,
    traceRouteCtlProbesPerHop,
    traceRouteCtlPort,
    traceRouteCtlMaxTtl,
    traceRouteCtlDSField,
    traceRouteCtlSourceAddressType,
    traceRouteCtlSourceAddress,
    traceRouteCtlIfIndex,
    traceRouteCtlMiscOptions,
    traceRouteCtlMaxFailures,
    traceRouteCtlDontFragment,
    traceRouteCtlInitialTtl,
    traceRouteCtlFrequency,
    traceRouteCtlStorageType,
    traceRouteCtlAdminStatus,
    traceRouteCtlMaxRows,
    traceRouteCtlTrapGeneration,
    traceRouteCtlDescr,
    traceRouteCtlCreateHopsEntries,
    traceRouteCtlType,
    traceRouteCtlRowStatus,
    traceRouteResultsOperStatus,
    traceRouteResultsCurHopCount,
    traceRouteResultsCurProbeCount,
    traceRouteResultsIpTgtAddrType,
    traceRouteResultsIpTgtAddr,
    traceRouteResultsTestAttempts,
    traceRouteResultsTestSuccesses,
    traceRouteProbeHistoryHAddrType,
    traceRouteProbeHistoryHAddr,
    traceRouteProbeHistoryResponse,
    traceRouteProbeHistoryStatus,
    traceRouteProbeHistoryLastRC
}
STATUS current
DESCRIPTION
    "The group of objects that comprise the remote traceroute
    operation."
 ::= { traceRouteGroups 1 }

traceRouteTimeStampGroup OBJECT-GROUP
OBJECTS {
    traceRouteResultsLastGoodPath,
    traceRouteProbeHistoryTime
}
STATUS current
DESCRIPTION

```

```

    "The group of DateAndTime objects."
    ::= { traceRouteGroups 2 }

```

```

traceRouteNotificationsGroup NOTIFICATION-GROUP

```

```

    NOTIFICATIONS {
        traceRoutePathChange,
        traceRouteTestFailed,
        traceRouteTestCompleted
    }

```

```

    STATUS current

```

```

    DESCRIPTION

```

```

        "The notifications which are required to be supported by
        implementations of this MIB."

```

```

    ::= { traceRouteGroups 3 }

```

```

traceRouteHopsTableGroup OBJECT-GROUP

```

```

    OBJECTS {
        traceRouteHopsIpTgtAddressType,
        traceRouteHopsIpTgtAddress,
        traceRouteHopsMinRtt,
        traceRouteHopsMaxRtt,
        traceRouteHopsAverageRtt,
        traceRouteHopsRttSumOfSquares,
        traceRouteHopsSentProbes,
        traceRouteHopsProbeResponses,
        traceRouteHopsLastGoodProbe
    }

```

```

    STATUS current

```

```

    DESCRIPTION

```

```

        "The group of objects that comprise the traceRouteHopsTable."

```

```

    ::= { traceRouteGroups 4 }

```

```

END

```

4.3 DISMAN-NSLOOKUP-MIB

```

DISMAN-NSLOOKUP-MIB DEFINITIONS ::= BEGIN

```

```

IMPORTS

```

```

    MODULE-IDENTITY, OBJECT-TYPE,
    Unsigned32, mib-2, Integer32
        FROM SNMPv2-SMI
        FROM SNMPv2-TC
    MODULE-COMPLIANCE, OBJECT-GROUP
        FROM SNMPv2-CONF
    SnmpAdminString
        FROM SNMP-FRAMEWORK-MIB

```

```

InetAddressType, InetAddress
    FROM INET-ADDRESS-MIB;                -- RFC2851

lookupMIB MODULE-IDENTITY
    LAST-UPDATED "200009210000Z"          -- 21 September 2000
    ORGANIZATION "IETF Distributed Management Working Group"
    CONTACT-INFO
        "Kenneth White

        International Business Machines Corporation
        Network Computing Software Division
        Research Triangle Park, NC, USA

        E-mail: wkenneth@us.ibm.com"
    DESCRIPTION
        "The Lookup MIB (DISMAN-NSLOOKUP-MIB) enables determination
        of either the name(s) corresponding to a host address or of
        the address(es) associated with a host name at a remote host."

    -- Revision history

    REVISION      "200009210000Z"          -- 21 September 2000
    DESCRIPTION
        "Initial version, published as RFC 2925."

 ::= { mib-2 82 }

-- Top level structure of the MIB

lookupObjects      OBJECT IDENTIFIER ::= { lookupMIB 1 }
lookupConformance OBJECT IDENTIFIER ::= { lookupMIB 2 }

-- Simple Object Definitions

lookupMaxConcurrentRequests OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS       "requests"
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The maximum number of concurrent active lookup requests
        that are allowed within an agent implementation. A value
        of 0 for this object implies that there is no limit for
        the number of concurrent active requests in effect."
    DEFVAL { 10 }
    ::= { lookupObjects 1 }

```

```
lookupPurgeTime OBJECT-TYPE
    SYNTAX      Unsigned32 (0..86400)
    UNITS       "seconds"
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The amount of time to wait before automatically
        deleting an entry in the lookupCtlTable and any
        dependent lookupResultsTable entries
        after the lookup operation represented by an
        lookupCtlEntry has completed.

        An lookupCtlEntry is considered complete
        when its lookupCtlOperStatus object has a
        value of completed(3)."
    DEFVAL { 900 } -- 15 minutes as default
    ::= { lookupObjects 2 }

-- Lookup Control Table

lookupCtlTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LookupCtlEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Defines the Lookup Control Table for providing
        the capability of performing a lookup operation,
        gethostbyname or gethostbyaddr, from a remote host."
    ::= { lookupObjects 3 }

lookupCtlEntry OBJECT-TYPE
    SYNTAX      LookupCtlEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Defines an entry in the lookupCtlTable.  A
        lookupCtlEntry is initially indexed by
        lookupCtlOwnerIndex, which is of type SnmpAdminString,
        a textual convention that allows for use of the SNMPv3
        View-Based Access Control Model (RFC 2575 [11], VACM)
        and also allows an management application to identify
        its entries.  The second index element,
        lookupCtlOperationName, enables the same
        lookupCtlOwnerIndex entity to have multiple outstanding
        requests.

        The value of lookupCtlTargetAddressType determines which
        lookup function to perform.  Specification of dns(16)
```

as the value of this index implies that the gethostbyname function should be performed to determine the numeric addresses associated with a symbolic name via lookupResultsTable entries. Use of a value of either ipv4(1) or ipv6(2) implies that the gethostbyaddr function should be performed to determine the symbolic name(s) associated with a numeric address at a remote host."

```
INDEX {
    lookupCtlOwnerIndex,
    lookupCtlOperationName
}
 ::= { lookupCtlTable 1 }
```

LookupCtlEntry ::=

```
SEQUENCE {
    lookupCtlOwnerIndex          SnmpAdminString,
    lookupCtlOperationName      SnmpAdminString,
    lookupCtlTargetAddressType  InetAddressType,
    lookupCtlTargetAddress      InetAddress,
    lookupCtlOperStatus         INTEGER,
    lookupCtlTime               Unsigned32,
    lookupCtlRc                 Integer32,
    lookupCtlRowStatus          RowStatus
}
```

lookupCtlOwnerIndex OBJECT-TYPE

SYNTAX SnmpAdminString (SIZE(0..32))

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"To facilitate the provisioning of access control by a security administrator using the View-Based Access Control Model (RFC 2575, VACM) for tables in which multiple users may need to independently create or modify entries, the initial index is used as an 'owner index'. Such an initial index has a syntax of SnmpAdminString, and can thus be trivially mapped to a securityName or groupName as defined in VACM, in accordance with a security policy.

When used in conjunction with such a security policy all entries in the table belonging to a particular user (or group) will have the same value for this initial index. For a given user's entries in a particular table, the object identifiers for the information in these entries will have the same subidentifiers (except for the 'column' subidentifier) up to the end of the encoded owner index. To configure VACM to permit access to this

portion of the table, one would create vacmViewTreeFamilyTable entries with the value of vacmViewTreeFamilySubtree including the owner index portion, and vacmViewTreeFamilyMask 'wildcarding' the column subidentifier. More elaborate configurations are possible."

```
::= { lookupCtlEntry 1 }
```

lookupCtlOperationName OBJECT-TYPE

SYNTAX SnmpAdminString (SIZE(0..32))

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"The name of a lookup operation. This is locally unique, within the scope of an lookupCtlOwnerIndex."

```
::= { lookupCtlEntry 2 }
```

lookupCtlTargetAddressType OBJECT-TYPE

SYNTAX InetAddressType

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"Specifies the type of address for either performing a gethostbyname or a gethostbyaddr function at a remote host. Specification of dns(16) as the value for this object means that the gethostbyname function should be performed to return one or more numeric addresses. Use of a value of either ipv4(1) or ipv6(2) means that the gethostbyaddr function should be used to return the symbolic names associated with a remote host."

```
::= { lookupCtlEntry 3 }
```

lookupCtlTargetAddress OBJECT-TYPE

SYNTAX InetAddress

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"Specifies the address used for a resolver lookup at a remote host. The corresponding lookupCtlAddressType objects determines its type as well as the function that can be requested.

A value for this object MUST be set prior to transitioning its corresponding lookupCtlEntry to active(1) via lookupCtlRowStatus."

```
::= { lookupCtlEntry 4 }
```

lookupCtlOperStatus OBJECT-TYPE

```

SYNTAX      INTEGER {
                notStarted(2), -- operation has not started
                completed(3)   -- operation is done
            }
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "Reflects the operational state of an lookupCtlEntry:

        enabled(1)    - Operation is active.
        notStarted(2) - Operation has not been enabled.
        completed(3)  - Operation has completed.

    An operation is automatically enabled(1) when its
    lookupCtlRowStatus object is transitioned to active(1)
    status.  Until this occurs lookupCtlOperStatus MUST
    report a value of notStarted(2).  After the lookup
    operation completes (success or failure) the value
    for lookupCtlOperStatus MUST be transitioned to
    completed(3)."
```

::= { lookupCtlEntry 5 }

```

lookupCtlTime OBJECT-TYPE
SYNTAX      Unsigned32
UNITS       "milliseconds"
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "Reports the number of milliseconds that a lookup
    operation required to be completed at a remote host.
    Completed means operation failure as well as
    success."
```

::= { lookupCtlEntry 6 }

```

lookupCtlRc OBJECT-TYPE
SYNTAX      Integer32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The system specific return code from a lookup
    operation.  All implementations MUST return a value
    of 0 for this object when the remote lookup
    operation succeeds.  A non-zero value for this
    objects indicates failure.  It is recommended that
    implementations that support errno use it as the
    value of this object to aid a management
    application in determining the cause of failure."
```

::= { lookupCtlEntry 7 }

lookupCtlRowStatus OBJECT-TYPE

SYNTAX RowStatus
 MAX-ACCESS read-create
 STATUS current

DESCRIPTION

"This object allows entries to be created and deleted in the lookupCtlTable.

A remote lookup operation is started when an entry in this table is created via an SNMP SET request and the entry is activated. This occurs by setting the value of this object to CreateAndGo(4) during row creation or by setting this object to active(1) after the row is created.

A value MUST be specified for lookupCtlTargetAddress prior to a transition to active(1) state being accepted.

A remote lookup operation starts when its entry first becomes active(1). Transitions in and out of active(1) state have no effect on the operational behavior of a remote lookup operation, with the exception that deletion of an entry in this table by setting its RowStatus object to destroy(6) will stop an active remote lookup operation.

The operational state of a remote lookup operation can be determined by examination of its lookupCtlOperStatus object."

REFERENCE

"See definition of RowStatus in RFC 2579, 'Textual Conventions for SMIV2.'"

::= { lookupCtlEntry 8 }

-- Lookup Results Table

lookupResultsTable OBJECT-TYPE

SYNTAX SEQUENCE OF LookupResultsEntry
 MAX-ACCESS not-accessible
 STATUS current

DESCRIPTION

"Defines the Lookup Results Table for providing the capability of determining the results of a operation at a remote host.

One or more entries are added to the lookupResultsTable when a lookup operation, as reflected by an lookupCtlEntry, completes successfully. All entries related to a successful lookup operation MUST be added to the lookupResultsTable at the same time that the associating lookupCtlOperStatus object is transitioned to completed(2).

The number of entries added depends on the results determined for a particular lookup operation. All entries associated with an lookupCtlEntry are removed when the lookupCtlEntry is deleted.

A remote host can be multi-homed and have more than one IP address associated with it (gethostbyname results) and/or it can have more than one symbolic name (gethostbyaddr results).

The gethostbyaddr function is called with a host address as its parameter and is used primarily to determine a symbolic name to associate with the host address. Entries in the lookupResultsTable MUST be made for each host name returned. The official host name MUST be assigned a lookupResultsIndex of 1.

The gethostbyname function is called with a symbolic host name and is used primarily to retrieve a host address. If possible the primary host address SHOULD be assigned a lookupResultsIndex of 1."

```
::= { lookupObjects 4 }
```

```
lookupResultsEntry OBJECT-TYPE
```

```
SYNTAX      LookupResultsEntry
```

```
MAX-ACCESS  not-accessible
```

```
STATUS      current
```

```
DESCRIPTION
```

```
"Defines an entry in the lookupResultsTable. The first two index elements identify the lookupCtlEntry that a lookupResultsEntry belongs to. The third index element selects a single lookup operation result."
```

```
INDEX {
```

```
    lookupCtlOwnerIndex,
```

```
    lookupCtlOperationName,
```

```
        lookupResultsIndex
    }
 ::= { lookupResultsTable 1 }

LookupResultsEntry ::=
SEQUENCE {
    lookupResultsIndex      Unsigned32,
    lookupResultsAddressType InetAddressType,
    lookupResultsAddress    InetAddress
}

lookupResultsIndex OBJECT-TYPE
SYNTAX      Unsigned32 (1..'ffffffff'h)
MAX-ACCESS not-accessible
STATUS      current
DESCRIPTION
    "Entries in the lookupResultsTable are created when
    the result of a lookup operation is determined.

    Entries MUST be stored in the lookupResultsTable in
    the order that they are retrieved. Values assigned
    to lookupResultsIndex MUST start at 1 and increase
    in order."
 ::= { lookupResultsEntry 1 }

lookupResultsAddressType OBJECT-TYPE
SYNTAX      InetAddressType
MAX-ACCESS read-only
STATUS      current
DESCRIPTION
    "Indicates the type of result of a remote lookup
    operation. A value of unknown(0) implies that
    either the operation hasn't been started or that
    it has failed."
 ::= { lookupResultsEntry 2 }

lookupResultsAddress OBJECT-TYPE
SYNTAX      InetAddress
MAX-ACCESS read-only
STATUS      current
DESCRIPTION
    "Reflects a result for a remote lookup operation
    as per the value of lookupResultsAddressType."
 ::= { lookupResultsEntry 3 }

-- Conformance information
-- Compliance statements
```

```

lookupCompliances OBJECT IDENTIFIER ::= { lookupConformance 1 }
lookupGroups      OBJECT IDENTIFIER ::= { lookupConformance 2 }

-- Compliance statements

lookupCompliance MODULE-COMPLIANCE
  STATUS current
  DESCRIPTION
    "The compliance statement for the DISMAN-NSLOOKUP-MIB."
  MODULE -- this module
    MANDATORY-GROUPS {
      lookupGroup
    }

    OBJECT lookupMaxConcurrentRequests
    MIN-ACCESS read-only
    DESCRIPTION
      "The agent is not required to support SET
      operations to this object."

    OBJECT lookupPurgeTime
    MIN-ACCESS read-only
    DESCRIPTION
      "The agent is not required to support a SET
      operation to this object."
  ::= { lookupCompliances 1 }

-- MIB groupings

lookupGroup OBJECT-GROUP
  OBJECTS {
    lookupMaxConcurrentRequests,
    lookupPurgeTime,
    lookupCtlOperStatus,
    lookupCtlTargetAddressType,
    lookupCtlTargetAddress,
    lookupCtlTime,
    lookupCtlRc,
    lookupCtlRowStatus,
    lookupResultsAddressType,
    lookupResultsAddress
  }
  STATUS current
  DESCRIPTION
    "The group of objects that comprise the remote
    Lookup operation."
  ::= { lookupGroups 1 }

```

END

5.0 Security Considerations

Certain management information in the MIBs defined by this document may be considered sensitive in some network environments. Therefore, authentication of received SNMP requests and controlled access to management information SHOULD be employed in such environments. The method for this authentication is a function of the SNMP Administrative Framework, and has not been expanded by this MIB.

To facilitate the provisioning of access control by a security administrator using the View-Based Access Control Model (VACM) defined in RFC 2575 [11] for tables in which multiple users may need to independently create or modify entries, the initial index is used as an "owner index". Such an initial index has a syntax of `SnmpAdminString`, and can thus be trivially mapped to a `securityName` or `groupName` as defined in VACM, in accordance with a security policy.

All entries in related tables belonging to a particular user will have the same value for this initial index. For a given user's entries in a particular table, the object identifiers for the information in these entries will have the same subidentifiers (except for the "column" subidentifier) up to the end of the encoded owner index. To configure VACM to permit access to this portion of the table, one would create `vacmViewTreeFamilyTable` entries with the value of `vacmViewTreeFamilySubtree` including the owner index portion, and `vacmViewTreeFamilyMask` "wildcarding" the column subidentifier. More elaborate configurations are possible. The VACM access control mechanism described above provides control.

In general, both the ping and traceroute functions when used excessively are considered a form of system attack. In the case of ping sending a system requests too often can negatively effect its performance or attempting to connect to what is supposed to be an unused port can be very unpredictable. Excessive use of the

traceroute capability can like ping negatively affect system performance. In insecure environments it is RECOMMENDED that the MIBs defined within this memo not be supported.

6.0 Intellectual Property

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

7.0 Acknowledgments

This document is a product of the DISMAN Working Group.

8.0 References

- [1] Case, J., Fedor, M., Schoffstall, M. and J. Davin, "Simple Network Management Protocol", STD 15, RFC 1157, May 1990.
- [2] Postel, J., "Echo Protocol", STD 20, RFC 862, May 1983.
- [3] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.
- [4] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.
- [5] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.

- [6] Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1905, January 1996.
- [7] Harrington D., Presuhn, R. and B. Wijnen, "An Architecture for Describing SNMP Management Frameworks", RFC 2571, April 1999.
- [8] Case, J., Harrington D., Presuhn, R. and B. Wijnen, "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)", RFC 2572, April 1999.
- [9] Levi D., Meyer, P. and B. Stewart, "SNMPv3 Applications", RFC 2573, April 1999.
- [10] Blumenthal, U. and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", RFC 2574, April 1999.
- [11] Wijnen, B., Presuhn, R. and K. McCloghrie, "View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)", RFC 2575, April 1999.
- [12] Hovey, R. and S. Bradner, "The Organizations Involved in the IETF Standards Process", BCP 11, RFC 2028, October 1996.
- [13] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [14] Rose, M. and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based Internets", RFC 1155, May 1990.
- [15] Rose, M. and K. McCloghrie, "Concise MIB Definitions", RFC 1212, March 1991.
- [16] Rose, M., "A Convention for Defining Traps for use with the SNMP", RFC 1215, March 1991.
- [17] Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Introduction to Community-based SNMPv2", RFC 1901, January 1996.
- [18] Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1906, January 1996.
- [19] Bradner, S., "The Internet Standards Process -- Revision 3", RFC 2026, BCP 9, October 1996.

- [20] Postel, J., "Internet Control Message Protocol", RFC 792, September 1981.
- [21] Nichols, K., Blake, S., Baker, F. and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.
- [22] Daniele, M., Haberman, B., Routhier, S. and J. Schoenwaelder, "Textual Conventions for Internet Network Addresses", RFC 2851, June 2000.
- [23] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, June 2000.

9.0 Author's Address

Kenneth D. White
Dept. BRQA/Bldg. 501/G114
IBM Corporation
P.O.Box 12195
3039 Cornwallis
Research Triangle Park, NC 27709, USA

EMail: wkenneth@us.ibm.com

10. Full Copyright Statement

Copyright (C) The Internet Society (2000). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.

