Network Working Group
Request for Comments: 3201
Category: Standards Track

R. Steinberger
Paradyne Networks
O. Nicklass
RAD Data Communications Ltd.

January 2002

Definitions of Managed Objects for Circuit to Interface Translation

Status of this Memo

Copyright Notice

Abstract

This memo defines an extension of the Management Information Base (MIB) for use with network management protocols in TCP/IP-based internets. In particular, it defines objects for managing the insertion of interesting Circuit Interfaces into the ifTable. This is important for circuits that must be used within other MIB modules which require an ifEntry. It allows for integrated monitoring of circuits as well as routing to circuits using unaltered, pre-existing MIB modules.

Table of Contents

1. The SNMP Management Framework

The SNMP Management Framework presently consists of five major components:

- An overall architecture, described in RFC 2571 [1].

- 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 [2], STD 16, RFC 1212 [3] and RFC 1215 [4]. The second version, called SMIv2, is described in STD 58, RFC 2578 [5], RFC 2579 [6] and RFC 2580 [7].

- Message protocols for transferring management information. The first version of the SNMP message protocol is called SNMPv1 and described in STD 15, RFC 1157 [8]. A second version of the SNMP message protocol, which is not an Internet standards track protocol, is called SNMPv2c and described in RFC 1901 [9] and RFC 1906 [10]. The third version of the message protocol is called SNMPv3 and described in RFC 1906 [10], RFC 2572 [11] and RFC 2574 [12].

- Protocol operations for accessing management information. The first set of protocol operations and associated PDU formats is described in STD 15, RFC 1157 [8]. A second set of protocol operations and associated PDU formats is described in RFC 1905 [13].

- o A set of fundamental applications described in RFC 2573 [14] and the view-based access control mechanism described in RFC 2575 [15].

A more detailed introduction to the current SNMP Management Framework can be found in RFC 2570 [16].

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

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

2. Conventions

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, NOT RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in RFC 2119 [21].

3. Overview

This MIB module addresses the concept of inserting circuits, which are potentially virtual, into the ifTable. There are multiple reasons to allow circuits to be added to the ifTable. The most prevalent of which are the standard routing MIB tables such as the ipCidrRouteTable (IP-FORWARD-MIB) and the ipNetToMediaTable (IP-MIB) act on the ifIndex and the RMON MIBs (RMON-MIB and RMON2-MIB as defined in RFC 2819 [23] and RFC 2021 [19]) require the use of an ifIndex a DataSource.

There is a further need to potentially monitor or manage a circuit based on the directional flow of traffic going through it. For instance, monitoring of protocols passed on a circuit using RMON-II (RFC 2021 [19]) does not currently capture the direction of the flow. This MIB module provides the capability to define an interface based on the specific direction of the flow.

This section provides an overview and background of how to use this MIB module.

3.1. Circuit Concepts

There are multiple MIB modules that define circuits. Three commonly used MIB modules are FRAME-RELAY-DTE-MIB (RFC 2115) [20], FRNETSERV-MIB (RFC 2954) [18], and ATM-MIB (RFC 2515) [22]. These define management objects for frame relay DTEs, frame relay services, and ATM respectively. Each of these MIB modules contain the ability to add or delete circuits; however, none create a specific ifEntry for a circuit. The reason for this is that there are potentially multiple circuits and not every circuit needs to be managed as an individual interface. For example, not every circuit on a device needs to be monitored with RMON and not every circuit needs to be included as an individual circuit for routing. Further, the Interfaces Group MIB (RFC 2863) [17] strongly recommends that conceptual rows not be added to the ifTable for virtual circuits.

The rationale for creating conceptual rows in the ifTable for these circuits is that there is a need for their use in either management of routing or monitoring of data. Both of these functions require mapping to an ifIndex.

This MIB module is designed such that only those circuits that require an ifIndex need be added to the ifTable. This prevents over-populating the ifTable with useless or otherwise unused indices.

While this document often refers to ATM and frame relay, it is not specifically designed for only those types of circuits. Any circuit that is defined in a MIB module but does not have its own ifIndex MAY be added with this MIB module.

3.2. Theory of Operation

3.2.1. Creation Process

In some cases, devices will automatically populate the rows of ciCircuitTable as circuits are created or discovered. However, in many cases, it may be necessary for a network manager to manually create rows.

Manual creation of rows requires the following steps:

1) Locate or create the circuit that is to be added on the device.

2) Create a row in ciCircuitTable for each flow type that is required.

The first step above requires some knowledge of the circuits that exist on a device. Typically, logical ports have entries in the

ifTable. If, for example, the ifType for the logical port is frameRelay(32), the circuits can be located in the frCircuitTable of the Frame Relay DTE MIB (FRAME-RELAY-DTE-MIB) [18]. If, as another example, the ifType for the logical port is frameRelayService(44), the circuits can be located in the frPVCEndptTable of the Frame Relay Service MIB (FRNETSERV-MIB) [20]. If, as a final example, the ifType for the logical port is aal5(49), the circuits can be located in the aal5VccTable of the ATM MIB (ATM-MIB) [22]. An entry describing the circuit MUST exist in some table prior to creating a row in ciCircuitTable. The object identifier that MUST be used in the circuit definition is the lexicographically smallest accessible OID that fully describes the the circuit.

3.2.2. Destruction Process

3.2.2.1. Manual Row Destruction

Manual row destruction is straight forward. Any row can be destroyed and the resources allocated to it are freed by setting the value of its status object (ciCircuitStatus) to destroy(6). It should be noted that when ciCircuitStatus is set to destroy(6) all associated rows in the ifTable and in ciIfMapTable will also be destroyed. This process MAY trigger further row destruction in other tables as well.

3.2.2.2. Automatic Row Destruction

Rows in the tables MAY be destroyed automatically based on the existence of the circuit on which they rely. When a circuit no longer exists in the device, the data in the tables has no relation to anything known on the network. For this reason, rows MUST be removed from this table as soon as it is discovered that the associated circuits no longer exist. The effects of automatic row destruction are the same as manual row destruction.

3.2.3. Modification Process

Since no objects in the MIB module can be changed once rows are active, there are no modification caveats.

3.2.4. Persistence of Data

Each row in the tables of this MIB module relies on information from other MIB modules. The rules for persistence of the data SHOULD follow the same rules as those of the underlying MIB module. For example, if the circuit defined by ciCircuitObject would normally be stored in non-volatile memory, then the ciCircuitEntry SHOULD also be non-volatile.

4. Relation to Other MIB Modules

4.1. Frame Relay DTE MIB

There is no required relation to the Frame Relay DTE MIB beyond the fact that rows in the frCircuitTable MAY be referenced. However, if frCircuitLogicalIfIndex is being used to represent the same information as a ciCircuitEntry with a value of ciCircuitFlow equal to both(3), the implementation MAY use the same ifIndex.

4.2. Frame Relay Service MIB

There is no explicit relation to the Frame Relay Service MIB beyond the fact that a rows in the frPVCEndptTable MAY be referenced.

4.3. ATM MIB

There is no explicit relation to the ATM MIB beyond the fact that rows in multiple tables may be referenced.

4.4. Interfaces Group MIB

4.4.1. Interfaces Table (ifTable, ifXtable)

The following specifies how the Interfaces Group defined in the IF-MIB will be used for the management of interfaces created by this MIB module.

Values of specific ifTable objects for circuit interfaces are as follows:

Object Name	Value of Object
ifIndex	Each entry in the circuit table is represented by an ifEntry. The value of ifIndex is defined by the agent such that it complies with any internal numbering scheme.
ifType	The value of ifType is specific to the type of circuit desired. For example, the value for frame relay virtual circuits is frDlciEndPt(193) and the value for ATM virtual circuits is atmVciEndPt(194). If the circuit is to be used in RMON, propVirtual(53) SHOULD NOT be used.

ifMtu	Set to the size in octets of the largest packet, frame or PDU supported on the circuit. If this is not known to the ifMtu object shall be set to zero. If the circuit is not modeled as a packet-oriented interface, this object SHOULD NOT be supported and result in noSuchInstance.
ifSpeed	The peak bandwidth in bits per second available for use. This will equal either the ifSpeed of the logical link if policing is not enforced or the maximum information rate otherwise. If neither is known, the ifSpeed object shall be set to zero.
ifPhysAddress	This will always be an octet string of zero length.
ifInOctets	The number of octets received by the network (ingress) for this circuit. This counter should count only octets included the header information and user data. If the device does not support statistics on the circuit, this object MUST NOT be supported and result in noSuchInstance.
ifInUcastPkts	The unerrored number of frames, packets or PDUs received by the network (ingress) for this circuit. If the device does not support statistics on the circuit, this object MUST NOT be supported and result in noSuchInstance.
ifInDiscards	The number of received frames, packets or PDUs for this circuit discarded due to ingress buffer congestion and traffic policing. If the device does not support statistics on the circuit, this object MUST NOT be supported and result in noSuchInstance.
ifInErrors	The number of received frames, packets or PDUs for this circuit that are discarded because of an error. If the device does not support statistics on the circuit, this object MUST NOT be supported and result in noSuchInstance.
ifOutOctets	The number of octets sent by the network (egress) for this circuit. This counter should count only octets included the header information and user data. If the device does not support statistics on the circuit, this object MUST NOT be supported and result in noSuchInstance.

ifOutUcastpkts The number of unerrored frames, packets or PDUs sent by the network (egress) for this circuit. If the device does not support statistics on the circuit, this object MUST NOT be supported and result in noSuchInstance.

ifOutDiscards The number of frames, packets or PDUs discarded in the egress direction for this circuit. Possible reasons are as follows: policing, congestion. If the device does not support statistics on the circuit, this object MUST NOT be supported and result in noSuchInstance.

ifOutErrors The number of frames, packets or PDUs discarded for this circuit in the egress direction because of an error. If the device does not support statistics on the circuit, this object MUST NOT be supported and result in noSuchInstance.

ifInBroadcastPkts
If the device does not support statistics on the circuit, this object MUST NOT be supported and result in noSuchInstance.

ifOutBroadcastPkts
If the device does not support Broadcast packets on the circuit, this object should not be supported and result in noSuchInstance.

ifLinkUpDownTrapEnable
Set to false(2). Circuits often have a predefined notification mechanism. In such instances, the number of notification sent would be doubled if this were enabled.

ifPromiscuousMode
Set to false(2). If the circuit is not modeled as a packet-oriented interface, this object SHOULD NOT be supported and result in noSuchInstance.

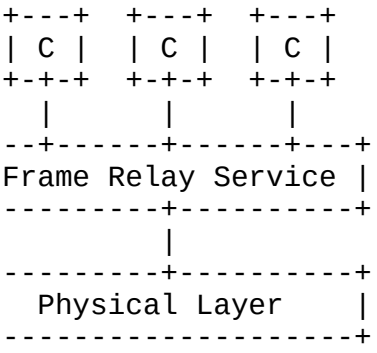
ifConnectorPresent
Set to false(2).

All other values are supported as stated in the IF-MIB documentation.

4.4.2. Stack Table (ifStackTable)

This section describes by example how to use ifStackTable to represent the relationship between circuit and logical link interfaces.

Example 1: Circuits (C) on a frame relay logical link.



The assignment of the index values could for example be (for a V35 physical interface):

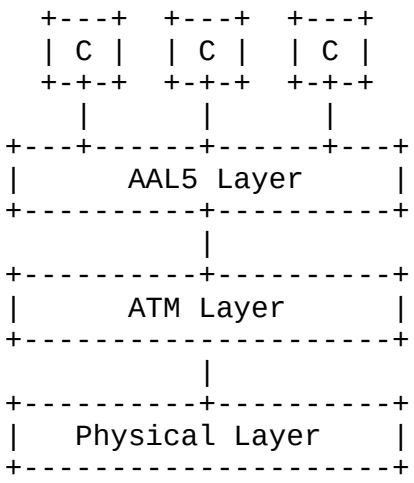
ifIndex	Description	
1	frDlciEndPt	(type 193)
2	frDlciEndPt	(type 193)
3	frDlciEndPt	(type 193)
4	frameRelayService	(type 44)
5	v35	(type 33)

The ifStackTable is then used to show the relationships between each interface.

HigherLayer	LowerLayer
0	1
0	2
0	3
1	4
2	4
3	4
4	5
5	0

In the above example the frame relay logical link could just as easily be of type frameRelay(32) instead.

Example 2: Circuits (C) on a AAL5 logical link.



The assignment of the index values could for example be (for a DS3 physical interface):

ifIndex	Description
1	atmVciEndPt (type 194)
2	atmVciEndPt (type 194)
3	atmVciEndPt (type 194)
4	aal5 (type 49)
5	atm (type 37)
6	ds3 (type 30)

The ifStackTable is then used to show the relationships between each interface.

HigherLayer	LowerLayer
0	1
0	2
0	3
1	4
2	4
3	4
4	5
5	6
6	0

4.5. Other MIB Modules

There is no explicit relation to any other media specific MIB module beyond the fact that rows in multiple tables may be referenced.

5. Structure of the MIB Module

The CIRCUIT-IF-MIB consists of the following components:

- ciCircuitTable
- ciIfMapTable

Refer to the compliance statement defined within for a definition of what objects MUST be implemented.

5.1. ciCircuitTable

The ciCircuitTable is the central control table for operations of the Circuit Interfaces MIB. It provides a means of mapping a circuit to its ifIndex as well as forcing the insertion of an ifIndex into the ifTable. The agent is responsible for managing the ifIndex itself such that no device dependent indexing scheme is violated.

A row in this table MUST exist in order for a row to exist in any other table in this MIB module.

5.2. ciIfMapTable

This table maps the ifIndex back to the circuit that it is associated with.

6. Object Definitions

```
CIRCUIT-IF-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE,
    mib-2, Gauge32                          FROM SNMPv2-SMI
    TEXTUAL-CONVENTION, RowStatus,
    TimeStamp, RowPointer, StorageType      FROM SNMPv2-TC
    MODULE-COMPLIANCE, OBJECT-GROUP         FROM SNMPv2-CONF
    ifIndex, InterfaceIndex                 FROM IF-MIB;

    circuitIfMIB MODULE-IDENTITY
        LAST-UPDATED "200201030000Z" -- January 3, 2002
        ORGANIZATION "IETF Frame Relay Service MIB Working Group"
        CONTACT-INFO
```

```
          "IETF Frame Relay Service MIB (frnetmib) Working Group

           WG Charter:    http://www.ietf.org/html.charters/
                                 frnetmib-charter.html
           WG-email:      frnetmib@sunroof.eng.sun.com
           Subscribe:     frnetmib-request@sunroof.eng.sun.com
           Email Archive: ftp://ftp.ietf.org/ietf-mail-archive/frnetmib

           Chair:      Andy Malis
                       Vivace Networks
           Email:      Andy.Malis@vivacenetworks.com

           WG editor:  Robert Steinberger
                       Paradyne Networks and
                       Fujitsu Network Communications
           Email:      robert.steinberger@fnc.fujitsu.com

           Co-author:  Orly Nicklass
                       RAD Data Communications Ltd.
           EMail:      Orly_n@rad.co.il"
       DESCRIPTION
           "The MIB module to allow insertion of selected circuit into
            the ifTable."
       REVISION "200201030000Z" -- January 3, 2002
       DESCRIPTION
           "Initial version, published as RFC 3201"
       ::= { mib-2 94 }

   -- Textual Conventions

   CiFlowDirection ::= TEXTUAL-CONVENTION
       STATUS  current
       DESCRIPTION
           "The direction of data flow thru a circuit.

                transmit(1) - Only transmitted data
                receive(2)  - Only received data
                both(3)     - Both transmitted and received data."
       SYNTAX  INTEGER {
                 transmit(1),
                 receive(2),
                 both(3)
               }

   ciObjects      OBJECT IDENTIFIER ::= { circuitIfMIB 1 }
   ciCapabilities OBJECT IDENTIFIER ::= { circuitIfMIB 2 }
   ciConformance  OBJECT IDENTIFIER ::= { circuitIfMIB 3 }
```

```
    -- The Circuit Interface Circuit Table
    --
    -- This table is used to define and display the circuits that
    -- are added to the ifTable.  It maps circuits to their respective
    -- ifIndex values.

    ciCircuitTable  OBJECT-TYPE
        SYNTAX      SEQUENCE OF CiCircuitEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "The Circuit Interface Circuit Table."
        ::= { ciObjects 1 }

    ciCircuitEntry OBJECT-TYPE
        SYNTAX      CiCircuitEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "An entry in the Circuit Interface Circuit Table."
        INDEX    { ciCircuitObject, ciCircuitFlow }
        ::= { ciCircuitTable 1 }

    CiCircuitEntry ::=
        SEQUENCE {
            --
            -- Index Control Variables
            --
            ciCircuitObject      RowPointer,
            ciCircuitFlow        CiFlowDirection,
            ciCircuitStatus      RowStatus,
            --
            -- Data variables
            --
            ciCircuitIfIndex     InterfaceIndex,
            ciCircuitCreateTime  TimeStamp,
            --
            -- Data Persistence
            --
            ciCircuitStorageType StorageType
        }

    ciCircuitObject OBJECT-TYPE
        SYNTAX      RowPointer
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "This value contains the RowPointer that uniquely
```

```
            describes the circuit that is to be added to this table.
            Any RowPointer that will force the size of OBJECT
            IDENTIFIER of the row to grow beyond the legal limit
            MUST be rejected.

            The purpose of this object is to point a network manager
            to the table in which the circuit was created as well as
            define the circuit on which the interface is defined.

            Valid tables for this object include the frCircuitTable
            from the Frame Relay DTE MIB(FRAME-RELAY-DTE-MIB), the
            frPVCEndptTable from the Frame Relay Service MIB
            (FRNETSERV-MIB), and the aal5VccTable from the ATM MIB
            (ATM MIB).  However, including circuits from other MIB
            tables IS NOT prohibited."
        ::= { ciCircuitEntry 1 }

    ciCircuitFlow OBJECT-TYPE
        SYNTAX      CiFlowDirection
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "The direction of data flow through the circuit for which
             the virtual interface is defined.  The following define
             the information that the virtual interface will report.

                transmit(1) - Only transmitted frames
                receive(2)  - Only received frames
                both(3)     - Both transmitted and received frames.

             It is recommended that the ifDescr of the circuit
             interfaces that are not both(3) SHOULD have text warning
             the operators that the particular interface represents
             only half the traffic on the circuit."
        ::= { ciCircuitEntry 2 }

    ciCircuitStatus OBJECT-TYPE
        SYNTAX      RowStatus
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
            "The status of the current row.  This object is
             used to add, delete, and disable rows in this
             table.  When the status changes to active(1), a row
             will also be added to the interface map table below
             and a row will be added to the ifTable.  These rows
             SHOULD not be removed until the status is changed
             from active(1).  The value of ifIndex for the row that
```

```
            is added to the ifTable is determined by the agent
            and MUST follow the rules of the ifTable.  The value
            of ifType for that interface will be frDlciEndPt(193)
            for a frame relay circuit, atmVciEndPt(194) for an
            ATM circuit, or another ifType defining the circuit
            type for any other circuit.

            When this object is set to destroy(6), the associated
            row in the interface map table will be removed and the
            ifIndex will be removed from the ifTable.  Removing
            the ifIndex MAY initiate a chain of events that causes
            changes to other tables as well.

            The rows added to this table MUST have a valid object
            identifier for ciCircuitObject.  This means that the
            referenced object must exist and it must be in a table
            that supports circuits.

            The object referenced by ciCircuitObject MUST exist
            prior to transitioning a row to active(1).  If at any
            point the object referenced by ciCircuitObject does not
            exist or the row containing it is not in the active(1)
            state, the status SHOULD either age out the row or
            report notReady(3).  The effects transitioning from
            active(1) to notReady(3) are the same as those caused
            by setting the status to destroy(6).

            Each row in this table relies on information from other
            MIB modules.  The rules persistence of data SHOULD follow
            the same rules as those of the underlying MIB module.
            For example, if the circuit defined by ciCircuitObject
            would normally be stored in non-volatile memory, then
            the row SHOULD also be non-volatile."
        ::= { ciCircuitEntry 3 }

    ciCircuitIfIndex OBJECT-TYPE
        SYNTAX      InterfaceIndex
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The ifIndex that the agent assigns to this row."
        ::= { ciCircuitEntry 4 }

    ciCircuitCreateTime OBJECT-TYPE
        SYNTAX      TimeStamp
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
```

```
            "This object returns the value of sysUpTime at the time
             the value of ciCircuitStatus last transitioned to
             active(1).  If ciCircuitStatus has never been active(1),
             this object SHOULD return 0."
        ::= { ciCircuitEntry 5 }

    ciCircuitStorageType OBJECT-TYPE
        SYNTAX      StorageType
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
            "The storage type used for this row."
    ::= { ciCircuitEntry 6 }

    -- The Circuit Interface Map Table
    --
    -- This table maps the ifIndex values that are assigned to
    -- rows in the circuit table back to the objects that define
    -- the circuits.

    ciIfMapTable  OBJECT-TYPE
        SYNTAX      SEQUENCE OF CiIfMapEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "The Circuit Interface Map Table."
        ::= { ciObjects 2 }

    ciIfMapEntry OBJECT-TYPE
        SYNTAX      CiIfMapEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "An entry in the Circuit Interface Map Table."
        INDEX    { ifIndex }
        ::= { ciIfMapTable 1 }

    CiIfMapEntry ::=
        SEQUENCE {
            --
            -- Mapped Object Variables
            --
            ciIfMapObject      RowPointer,
            ciIfMapFlow        CiFlowDirection
        }

    ciIfMapObject OBJECT-TYPE
        SYNTAX      RowPointer
```

```
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "This value contains the value of RowPointer that
             corresponds to the current ifIndex."
        ::= { ciIfMapEntry 1 }

    ciIfMapFlow   OBJECT-TYPE
        SYNTAX      CiFlowDirection
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The value contains the value of ciCircuitFlow that
             corresponds to the current ifIndex."
        ::= { ciIfMapEntry 2 }

    -- Change tracking metrics

    ciIfLastChange OBJECT-TYPE
        SYNTAX       TimeStamp
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "The value of sysUpTime at the most recent change to
             ciCircuitStatus for any row in ciCircuitTable."
        ::= { ciObjects 3 }

    ciIfNumActive      OBJECT-TYPE
        SYNTAX         Gauge32
        MAX-ACCESS     read-only
        STATUS         current
        DESCRIPTION
            "The number of active rows in ciCircuitTable."
        ::= { ciObjects 4 }

    -- Conformance Information

    ciMIBGroups      OBJECT IDENTIFIER ::= { ciConformance 1 }
    ciMIBCompliances OBJECT IDENTIFIER ::= { ciConformance 2 }

    --
    -- Compliance Statements
    --

    ciCompliance MODULE-COMPLIANCE
        STATUS  current
        DESCRIPTION
            "The compliance statement for SNMP entities
```

```
            which support of the Circuit Interfaces MIB module.
            This group defines the minimum level of support
            required for compliance."
        MODULE -- this module
            MANDATORY-GROUPS { ciCircuitGroup,
                               ciIfMapGroup,
                               ciStatsGroup }

            OBJECT      ciCircuitStatus
            SYNTAX      INTEGER { active(1) } -- subset of RowStatus
            MIN-ACCESS  read-only
            DESCRIPTION
               "Row creation can be done outside of the scope of
                the SNMP protocol.  If this object is implemented with
                max-access of read-only, then the only value that MUST
                be returned is active(1)."

            OBJECT      ciCircuitStorageType
            MIN-ACCESS  read-only
            DESCRIPTION
               "It is legal to support ciCircuitStorageType as read-
                only as long as the value reported in consistent
                with the actual storage mechanism employed within the
                agent."

    ::= { ciMIBCompliances 1 }

    --
    -- Units of Conformance
    --
    ciCircuitGroup  OBJECT-GROUP
       OBJECTS {
            ciCircuitStatus,
            ciCircuitIfIndex,
            ciCircuitCreateTime,
            ciCircuitStorageType
       }
       STATUS  current
       DESCRIPTION
           "A collection of required objects providing
            information from the circuit table."
       ::= { ciMIBGroups 1 }

    ciIfMapGroup OBJECT-GROUP
       OBJECTS {
            ciIfMapObject,
            ciIfMapFlow
       }
```

```
       STATUS  current
       DESCRIPTION
           "A collection of required objects providing
            information from the interface map table."
       ::= { ciMIBGroups 2 }

    ciStatsGroup OBJECT-GROUP
       OBJECTS {
            ciIfLastChange,
            ciIfNumActive
       }
       STATUS  current
       DESCRIPTION
           "A collection of statistical metrics used to help manage
            the ciCircuitTable."
       ::= { ciMIBGroups 3 }
END
```

7. Acknowledgments

This document was produced by the Frame Relay Service MIB Working Group.

8. References

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

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

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

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

[5] 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.

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

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

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

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

[10] 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.

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

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

[13] 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.

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

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

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

[17] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, June 2000.

[18] Rehbehn, K. and D. Fowler, "Definitions of Managed Objects for Frame Relay Service", RFC 2954, October 2000.

[19] Waldbusser, S., "Remote Network Monitoring Management Information Base Version 2 using SMIv2", RFC 2021, January 1997.

[20] Brown, C. and F. Baker, "Management Information Base for Frame Relay DTEs Using SMIv2", RFC 2115, September 1997.

[21] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[22] Tesink, K., "Definitions of Managed Objects for ATM Management", RFC 2515, February 1999.

[23] Waldbusser, S., "Remote Network Monitoring Management Information Base", RFC 2819, May 2000.

9. Security Considerations

There are a number of management objects defined in this MIB that have a MAX-ACCESS clause of read-write and/or read-create. Such objects may be considered sensitive or vulnerable in some network environments. The support for SET operations in a non-secure environment without proper protection can have a negative effect on network operations.

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

It is recommended that the implementers consider the security features as provided by the SNMPv3 framework. Specifically, the use of the User-based Security Model RFC 2274 [12] and the View-based Access Control Model RFC 2275 [15] is recommended.

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

10. IANA Considerations

New ifTypes defined specifically for use in this MIB module SHOULD be in the form of ***EndPt. This is similar to frDlciEndPt(193) and atmVciEndPt(194) which are already defined.

11. Authors' Addresses

Robert Steinberger
Fujitsu Network Communications
2801 Telecom Parkway
Richardson, TX 75082

Phone: 1-972-479-4739
EMail: robert.steinberger@fnc.fujitsu.com

Orly Nicklass, Ph.D
RAD Data Communications Ltd.
12 Hanechoshet Street
Tel Aviv, Israel 69710

Phone: 972 3 7659969
EMail: Orly_n@rad.co.il

12. Full Copyright Statement

Acknowledgement

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