

## Extensions to the Generic-Interface MIB

### Status of this Memo

This RFC contains definitions of managed objects used as experimental extensions to the generic interfaces structure of MIB-II. This memo is a product of the SNMP Working Group of the Internet Engineering Task Force (IETF). This RFC specifies an IAB standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "IAB Official Protocol Standards" for the standardization state and status of this protocol. Distribution of this memo is unlimited.

### Table of Contents

1. Abstract .....	1
2. The Network Management Framework.....	1
3. Objects .....	2
4. Overview .....	3
4.1 Generic Interface Extension Table .....	3
4.2 Generic Interface Test Table .....	3
4.3 Generic Receive Address Table .....	4
5. Definitions .....	5
6. Acknowledgements .....	14
7. References .....	15
8. Security Considerations.....	15
9. Author's Address.....	16

### 1. Abstract

This memo defines an experimental portion of the Management Information Base (MIB) for use with network management protocols in TCP/IP-based internets. In particular, it defines managed object types as experimental extensions to the generic interfaces structure of MIB-II.

### 2. The Network Management Framework

The Internet-standard Network Management Framework consists of three components. They are:

RFC 1155 which defines the SMI, the mechanisms used for describing and naming objects for the purpose of management. RFC 1212

defines a more concise description mechanism, which is wholly consistent with the SMI.

RFC 1156 which defines MIB-I, the core set of managed objects for the Internet suite of protocols. RFC 1213, defines MIB-II, an evolution of MIB-I based on implementation experience and new operational requirements.

RFC 1157 which defines the SNMP, the protocol used for network access to managed objects.

The Framework permits new objects to be defined for the purpose of experimentation and evaluation.

### 3. Objects

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are defined using the subset of Abstract Syntax Notation One (ASN.1) [7] defined in the SMI. In particular, each object has a name, a syntax, and an encoding. The name is an object identifier, an administratively assigned name, which specifies an object type. The object type together with an object instance serves to uniquely identify a specific instantiation of the object. For human convenience, we often use a textual string, termed the OBJECT DESCRIPTOR, to also refer to the object type.

The syntax of an object type defines the abstract data structure corresponding to that object type. The ASN.1 language is used for this purpose. However, the SMI [3] purposely restricts the ASN.1 constructs which may be used. These restrictions are explicitly made for simplicity.

The encoding of an object type is simply how that object type is represented using the object type's syntax. Implicitly tied to the notion of an object type's syntax and encoding is how the object type is represented when being transmitted on the network.

The SMI specifies the use of the basic encoding rules of ASN.1 [8], subject to the additional requirements imposed by the SNMP.

Section 5 contains the specification of all object types in this section of the MIB. The object types are defined using the conventions specified in the SMI, as amended by the extensions specified in [9].

## 4. Overview

The Internet Standard MIB [4,6] contains a group of management objects pertaining to a network device's generic network interface(s). These objects are generic in the sense that they apply to all network interfaces, irrespective of the type of communication media and protocols used on such interfaces. This has proved to be necessary but not sufficient; there are efforts underway to define additional MIB objects which are specific to particular media and lower-level (subnetwork-layer and below) protocol stacks.

However, some of these efforts have identified objects which are required (or at least useful), but are not specific to the interface-type on which the effort is focusing. In order to avoid redundancy, it is better that such objects be defined as extensions to the generic interface group, rather than defined in multiple specific-interface-type MIBs.

This memo defines the resultant extensions to the generic interface group. These extensions are spread over three tables: the generic Interface Extension table, the generic Interface Test table, and the generic Receive Address table.

### 4.1. Generic Interface Extension Table

This table consists of new objects applicable to all types of subnetwork interface.

### 4.2. Generic Interface Test Table

This section defines objects which allow a network manager to instruct an agent to test an interface for various faults. A few common types of tests are defined in this document but most will be defined elsewhere, dependent on the particular type of interface. After testing, the object `ifExtnsTestResult` can be read to determine the outcome. If an agent cannot perform the test, `ifExtnsTestResult` is set to so indicate. The object `ifExtnsTestCode` can be used to provide further test-specific or interface-specific (or even enterprise-specific) information concerning the outcome of the test. Only one test can be in progress on each interface at any one time. If one test is in progress when another test is invoked, the second test is rejected. Some agents may reject a test when a prior test is active on another interface.

When a test is invoked, the identity of the originator of the request and the request-id are saved by the agent in the objects `ifExtnsTestRequestId` and `ifExtnsTestCommunity`. These values remain set until the next test is invoked. In the (rare) event that the

invocation of tests by two network managers were to overlap, then there would be a possibility that the first test's results might be overwritten by the second test's results prior to the first results being read. This unlikely circumstance can be detected by a network manager retrieving `ifExtnsTestCommunity`, and `ifExtnsTestRequestId` at the same time as the test results are retrieved, and ensuring that the results are for the desired request.

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

Some tests may require the interface to be taken off-line or may even require the agent to be rebooted after completion of the test. In these circumstances, communication with the management station invoking the test may be lost until after completion of the test. The agent should make every effort to transmit a response to the request that invoked the test prior to losing communication. When the agent is restored to normal service, the results of the test are properly made available in the appropriate objects. Note that this requires that the `ifIndex` value assigned to an interface must be unchanged even if the test causes a reboot. An agent must reject any test for which it cannot, perhaps due to resource constraints, make available at least the minimum amount of information after that test completes.

#### 4.3. Generic Receive Address Table

This table contains objects relating to an interface's support for receiving packets/frames at more than one address on the same interface.

## 5. Definitions

```
RFC1229-MIB DEFINITIONS ::= BEGIN

--          Extensions to MIB-II's Generic Interface Table

IMPORTS
    experimental, Counter          FROM RFC1155-SMI
    DisplayString, PhysAddress     FROM RFC1213-MIB
    OBJECT-TYPE                   FROM RFC-1212;

ifExtensions OBJECT IDENTIFIER ::= { experimental 6 }

--      Generic Interface Extension Table
--
--      This group of objects is mandatory for all types of
--      subnetwork interface.

ifExtnsTable OBJECT-TYPE
    SYNTAX SEQUENCE OF IfExtnsEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A list of interfaces extension entries.
         The number of entries is given by the value
         of ifNumber, defined in [4,6]."
    ::= { ifExtensions 1 }

ifExtnsEntry OBJECT-TYPE
    SYNTAX IfExtnsEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "An extension to the interfaces entry,
         defined in [4,6], containing additional
         objects at the subnetwork layer and below
         for a particular interface."
    INDEX { ifExtnsIfIndex }
    ::= { ifExtnsTable 1 }

IfExtnsEntry ::=
    SEQUENCE {
        ifExtnsIfIndex
```

```

        INTEGER,
ifExtnsChipSet
        OBJECT IDENTIFIER,
ifExtnsRevWare
        DisplayString,
ifExtnsMulticastsTransmittedOks
        Counter,
ifExtnsBroadcastsTransmittedOks
        Counter,
ifExtnsMulticastsReceivedOks
        Counter,
ifExtnsBroadcastsReceivedOks
        Counter,
ifExtnsPromiscuous
        INTEGER
    }

ifExtnsIfIndex OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The value of this object identifies the
        interface for which this entry contains
        extended management information. The value
        of this object for a particular interface
        has the same value as the ifIndex object,
        defined in [4,6], for the same interface."
    ::= { ifExtnsEntry 1 }

ifExtnsChipSet OBJECT-TYPE
    SYNTAX OBJECT IDENTIFIER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "This object identifies the hardware chip
        set being used in the interface. The
        assignment of OBJECT IDENTIFIERS to various
        types of hardware chip sets is managed
        by the IANA. If the hardware chip set is
        unknown, the object identifier

        unknownChipSet OBJECT IDENTIFIER ::= { 0 0 }

        is returned. Note that unknownChipSet is a
        syntactically valid object identifier, and
        any conformant implementation of ASN.1 and
        the BER must be able to generate and

```

```
        recognize this value."
 ::= { ifExtnsEntry 2 }

ifExtnsRevWare OBJECT-TYPE
    SYNTAX DisplayString (SIZE (0..255))
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "An arbitrary octet string that describes
        the firmware version of this interface.
        It is intended that this should be human
        readable. It must only contain ASCII
        printable characters. Typically this
        will be the firmware version of the main
        interface software."
 ::= { ifExtnsEntry 3 }

ifExtnsMulticastsTransmittedOks OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The count of frames successfully
        transmitted to a subnetwork or link-layer
        multicast destination address other than a
        broadcast address. For a MAC layer protocol,
        this includes both Group and Functional
        addresses."
 ::= { ifExtnsEntry 4 }

ifExtnsBroadcastsTransmittedOks OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The count of frames successfully
        transmitted to a subnetwork or link-layer
        broadcast addresses. It does not include
        frames sent to a multicast address."
 ::= { ifExtnsEntry 5 }

ifExtnsMulticastsReceivedOks OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The count of frames successfully received
        that are directed to an active subnetwork
```

```

        or link-layer multicast address (for a MAC
        layer protocol, this includes both Group and
        Functional addresses). This does not include
        frames directed to a broadcast address, nor
        frames received with errors."
 ::= { ifExtnsEntry 6 }

ifExtnsBroadcastsReceivedOks OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The count of frames successfully received
        that are directed to a subnetwork or
        link-layer broadcast address. This does not
        include frames received with errors."
 ::= { ifExtnsEntry 7 }

ifExtnsPromiscuous OBJECT-TYPE
    SYNTAX INTEGER {
        true(1),
        false(2)
    }
    ACCESS read-only -- Note: agent implementors are
                     -- encouraged to extend this
                     -- access to read-write if that
                     -- makes sense in their agent.
    STATUS mandatory
    DESCRIPTION
        "This object has a value of false(2) if
        this interface only accepts packets/frames
        that are addressed to this station. This
        object has a value of true(1) when the
        station accepts all packets/frames
        transmitted on the media. The value
        true(1) is only legal on certain types of
        media. If legal, setting this object to a
        value of true(1) may require the interface
        to be reset before becoming effective."
 ::= { ifExtnsEntry 8 }

--
--     Generic Interface Test Table
--
-- This group of objects is optional, but if the table is
-- implemented, all objects in the table must be implemented.

```



ifExtnsTestTable OBJECT-TYPE

SYNTAX SEQUENCE OF IfExtnsTestEntry

ACCESS not-accessible

STATUS mandatory

DESCRIPTION

"This table contains one entry per interface."

::= { ifExtensions 2 }

ifExtnsTestEntry OBJECT-TYPE

SYNTAX IfExtnsTestEntry

ACCESS not-accessible

STATUS mandatory

DESCRIPTION

"An entry containing objects for invoking tests on an interface."

INDEX { ifExtnsTestIfIndex }

::= { ifExtnsTestTable 1 }

IfExtnsTestEntry ::=

SEQUENCE {

ifExtnsTestIfIndex

INTEGER,

ifExtnsTestCommunity

OCTET STRING,

ifExtnsTestRequestId

INTEGER,

ifExtnsTestType

OBJECT IDENTIFIER,

ifExtnsTestResult

INTEGER,

ifExtnsTestCode

OBJECT IDENTIFIER

}

ifExtnsTestIfIndex OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The value of this object identifies the interface for which this entry contains information on interface tests. The value of this object for a particular interface has the same value as the ifIndex object, defined in [4,6], for the same interface."

::= { ifExtnsTestEntry 1 }

```
ifExtnsTestCommunity OBJECT-TYPE
    SYNTAX OCTET STRING
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "This object contains the name of the SNMP
        authentication community [5] which was used
        to authenticate the SNMP Message which invoked
        the current or most recent test on this
        interface. If the authentication community
        is unknown or undefined, this value contains
        the zero-length string."
    ::= { ifExtnsTestEntry 2 }
```

```
ifExtnsTestRequestId OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "This object contains the value of the
        request-id field in the SNMP PDU [5] which
        invoked the current or most recent test on
        this interface. If the request-id is
        unknown or undefined, this value contains
        the value zero."
    ::= { ifExtnsTestEntry 3 }
```

```
ifExtnsTestType OBJECT-TYPE
    SYNTAX OBJECT IDENTIFIER
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "A control variable used to start and stop
        operator-initiated interface tests.

        Most OBJECT IDENTIFIER values assigned
        to tests are defined elsewhere, in associ-
        ation with specific types of interface.
        However, this document assigns a value for
        a full-duplex loopback test, and defines the
        special meanings of the subject identifier:

        noTest OBJECT IDENTIFIER ::= { 0 0 }
```

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

currently in progress, in which case the indicated test is initiated.

Note that noTest is a syntactically valid object identifier, and any conformant implementation of ASN.1 and BER must be able to generate and recognize this value.

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

```
::= { ifExtnsTestEntry 4 }
```

```
wellKnownTests OBJECT IDENTIFIER ::= { ifExtensions 4 }
```

```
-- full-duplex loopback test
```

```
testFullDuplexLoopBack OBJECT IDENTIFIER ::=
                                     { wellKnownTests 1 }
```

```
ifExtnsTestResult OBJECT-TYPE
```

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

```
    ACCESS  read-only
```

```
    STATUS  mandatory
```

```
    DESCRIPTION
```

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

```
::= { ifExtnsTestEntry 5 }
```

```
ifExtnsTestCode OBJECT-TYPE
```

```
    SYNTAX  OBJECT IDENTIFIER
```

```
    ACCESS  read-only
```

```
    STATUS  mandatory
```

```
    DESCRIPTION
```

"This object contains a code which contains more specific information on the test result, for example an error-code after a failed test. Error codes and other values this object may take are specific to the type of interface and/or test. However, one subject identifier:

```
testCodeUnknown OBJECT IDENTIFIER ::= { 0 0 }
```

for use if no additional result code is available.

Note that testCodeUnknown is a syntactically valid object identifier, and any conformant implementation of ASN.1 and the BER must be able to generate and recognize this value."

```
::= { ifExtnsTestEntry 6 }
```

```
-- Generic Receive Address Table
```

```
--
```

```
-- This group of objects is mandatory for all types of
-- interfaces which can receive packets/frames addressed to
-- more than one address.
```

```
ifExtnsRcvAddrTable OBJECT-TYPE
```

```
SYNTAX SEQUENCE OF IfExtnsRcvAddrEntry
```

```
ACCESS not-accessible
```

```
STATUS mandatory
```

```
DESCRIPTION
```

"This table contains an entry for each address (broadcast, multicast, or uni-cast) for which the system will receive packets/frames on a particular interface. When an interface is operating in promiscuous mode, entries are only required for those addresses for which the system would receive frames were it not operating in promiscuous mode."

```
::= { ifExtensions 3 }
```

```
ifExtnsRcvAddrEntry OBJECT-TYPE
```

```
SYNTAX IfExtnsRcvAddrEntry
```

```
ACCESS not-accessible
```

```
STATUS mandatory
```

```
DESCRIPTION
```

"A list of objects identifying an address for which the system will accept packets/

```
frames on a particular interface."
INDEX { ifExtnsRcvAddrIfIndex, ifExtnsRcvAddress }
 ::= { ifExtnsRcvAddrTable 1 }

IfExtnsRcvAddrEntry ::=
    SEQUENCE {
        ifExtnsRcvAddrIfIndex
            INTEGER,
        ifExtnsRcvAddress
            PhysAddress,
        ifExtnsRcvAddrStatus
            INTEGER
    }

ifExtnsRcvAddrIfIndex OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The value of ifIndex, defined in [4,6], of an
        interface which recognizes this entry's
        address."
    ::= { ifExtnsRcvAddrEntry 1 }

ifExtnsRcvAddress OBJECT-TYPE
    SYNTAX PhysAddress
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "An address for which the system will accept
        packets/frames on this entry's interface."
    ::= { ifExtnsRcvAddrEntry 2 }

ifExtnsRcvAddrStatus OBJECT-TYPE
    SYNTAX INTEGER {
        other(1),
        invalid(2),
        volatile(3),
        nonVolatile(4)
    }
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "This object has the value nonVolatile(4)
        for those entries in the table which are
        valid and will not be deleted by the next
        restart of the managed system.  Entries
        having the value volatile(3) are valid"
```

and exist, but have not been saved, so that will not exist after the next restart of the managed system. Entries having the value other(1) are valid and exist but are not classified as to whether they will continue to exist after the next restart. Entries having the value invalid(2) are invalid and do not represent an address for which an interface accepts frames.

Setting an object instance to one of the values other(1), volatile(3), or nonVolatile(4) causes the corresponding entry to exist or continue to exist, and to take on the respective status as regards the next restart of the managed system.

Setting an object instance to the value invalid(2) causes the corresponding entry to become invalid or cease to exist.

It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management stations must be prepared to receive tabular information from agents that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant ifExtnsRcvAddrStatus object instance."

```
DEFVAL { volatile }  
::= { ifExtnsRcvAddrEntry 3 }
```

END

## 6. Acknowledgements

Most of the MIB objects defined in this document were originally proposed as a part of a MIB for management of IEEE 802.5 Token Ring networks, as prepared by:

Eric B. Decker, cisco Systems, Inc., and  
Richard Fox, Synoptics Inc.

In addition, the comments of the following individuals are acknowledged:

James R. Davin, MIT-LCS  
Stan Froyd, ACC  
Frank Kastenholz, Racal Interlan

Dave Perkins, 3Com  
Marshall T. Rose, PSI  
Bob Stewart, Xyplex  
David Waitzman, BBN  
Wengyik Yeong, PSI

## 7. References

- [1] Cerf, V., "IAB Recommendations for the Development of Internet Network Management Standards", RFC 1052, NRI, April 1988.
- [2] Cerf, V., "Report of the Second Ad Hoc Network Management Review Group", RFC 1109, NRI, August 1989.
- [3] Rose M., and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based internets", RFC 1155, Performance Systems International, Hughes LAN Systems, May 1990.
- [4] McCloghrie K., and M. Rose, "Management Information Base for Network Management of TCP/IP-based internets", RFC 1156, Hughes LAN Systems, Performance Systems International, May 1990.
- [5] Case, J., Fedor, M., Schoffstall, M., and J. Davin, "Simple Network Management Protocol", RFC 1157, SNMP Research, Performance Systems International, Performance Systems International, MIT Laboratory for Computer Science, May 1990.
- [6] Rose M., Editor, "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", RFC 1213, Performance Systems International, March 1991.
- [7] Information processing systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1), International Organization for Standardization, International Standard 8824, December 1987.
- [8] Information processing systems - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Notation One (ASN.1), International Organization for Standardization, International Standard 8825, December 1987.
- [9] Rose, M., and K. McCloghrie, Editors, "Concise MIB Definitions", RFC 1212, Performance Systems International, Hughes LAN Systems, March 1991.

## 8. Security Considerations

Security issues are not discussed in this memo.

## 9. Author's Address

Keith McCloghrie  
Hughes LAN Systems, Inc.  
1225 Charleston Road  
Mountain View, CA 94043

Phone: (415) 966-7934

EMail: kzm@hls.com