Definitions of Managed Objects
for the HIPPI Interface Type

SNMP MIB Extension for HIPPI

September 28, 1992
Version 1

Jeff Miller (author)

Network Systems Corporation
7600 Boone Avenue North
Minneapolis, Minnesota 55428

Jeff.Miller@nsco.network.com

1. Status of this Memo

This document is an Internet Draft. Internet Drafts are working documents of the Internet Engineering Task Force (IETF), its Areas, and its Working Groups. Distribution of this memo is unlimited and encouraged. Please send comments to Jeff.Miller@nsco.network.com.

2. 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 objects for managing HIPPI objects. These objects are intended to be used in conjunction with the interface mib to fully monitor a HIPPI interface.

This memo does not specify a standard for the Internet community.

3. 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.

4. 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) [2] 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 [1] 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 [3], subject to the additional requirements imposed by the SNMP.

4.1. Format of Definitions

Section 6 contains the specification of all object types contained in this MIB module. The object types are defined using the conventions defined in the SMI, as amended by the extensions specified in RFC 1212.

5. Overview

These objects are used when the particular media being used to realize an interface is a HIPPI interface. At present, this applies to these values of the ifType variable in the Internet-standard MIB:

hippi (xx)

The definitions contained herein are based on the HIPPI specifications in ANSI X3.183-1991 (HIPPI-PH) and ANSI X3.222-199x (HIPPI-SC) [4, 5].

5.1. Textual Conventions

There are three counters in the HIPPI MIB which if implemented with the current SNMP counter definition of 32 bits would wrap within an 8 second time interval. In order to make these counters retain more useful information an attempt is made to define a counter capable of counting in excess of 2 Quintillion. This is accomplished by using two integers. One integer is defined as the “low” integer with a range of 0 to 999,999,999. The second integer is defined as the “high” integer with a range of 0 to 2147483647 (0x7FFFFFFF). When used as a counter, increment the “low” integer until it reaches 1,000,000,000 at which point increment the “high” counter and return the low counter to 0. Should the “high” integer ever reach 2147483647, it will return to 0 on its next increment. The reason for using the “low” integer limit of 999,999,999 is for decoding and implementation purposes.

It would be nice to group or some how datatype this large counter but the current constructs of SNMP do not allow for this. It may have been implemented as a specific sequence for a specific counter but there is no general sequence definition that could be used to create a datatype. Because of this, you will see the counter implemented as two integers that must be fetched separately and that are only bound by naming conventions.

The implication of using two integers is that it is impossible to read or set the counter as an autonomous operation. To reduce erroneous information it is suggested that reads and sets be performed as a series of operations either as a single message (preferred) or a series of messages. When setting the counter (mostly likely a reset to 0), first set the high integer, then the low integer, and finally the high integer once again. This type of operation could also be performed for reads so that by comparing the high integers it will be possible to determine if a wrap of the low integer occurred.

5.2. Structure of MIB

6. Definitions

```
RFC-HIPPI-MIB DEFINITIONS ::= BEGIN

IMPORTS
        Counter
                FROM RFC1155-SMI
        OBJECT-TYPE
                FROM RFC-1212
        transmission
                FROM RFC1213-MIB;


--  This MIB module uses the extended OBJECT-TYPE macro
--  as defined in [6].

--   HIPPI MIB module

hippi OBJECT IDENTIFIER ::= { transmission <number> }

--   textual conventions


--  It would be best to be able to define a datatype for a large counter
--  that could be implemented with two integers to provide for counters
--  that quickly grow beyond the 32 bit value defined in SNMP. Unfortunately
--  the current definitions do not allow for this so you will see the datatype
--  implemented as two separate integers (rather then one datatype) in the
--  MIB below.


--  The hippiTable object type consists of a hippiEntry
```

```
-- for every hippi object being supported.

hippiNumber OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "An integer value indicating the number of HippiEntry objects are in the
        hippiTable."
    ::= { hippi 1 }

hippiTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF HippiEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "The HIPPI table of entries"
    ::= { hippi 2 }

hippiEntry OBJECT-TYPE
    SYNTAX      HippiEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "An entry in the HIPPI table"
    INDEX       {hippiIndex }
    ::= { hippiTable 1 }
```

```
HippiEntry ::= SEQUENCE {
    hippiIndex
        INTEGER,
    hippiWordSize
        INTEGER,
    hippiEndPointType
        INTEGER,
    hippiWordCountHigh
        INTEGER,
    hippiWordCountLow
        INTEGER,
    hippiBurstCountHigh
        INTEGER,
    hippiBurstCountLow
        INTEGER,
    hippiPacketCountHigh
        INTEGER,
    hippiPacketCountLow
        INTEGER,
    hippiParityErrors
        Counter,
    hippiOverrunCount
        Counter,
    hippiInputPortRejectCount
        Counter,
    hippiCampOnDelayCount
        Counter,
    hippiDownlineRejectCount
        Counter,
    hippiUplineRejectCount
        Counter,
    hippiDownlineDisconnectCount
        Counter,
    hippiUplineDisconnectCount
        Counter,
    hippiLastIField
        OCTET STRING,
    hippiConnectState
```

```
        INTEGER,
    hippiLastErrorType
        INTEGER,
    hippiLastErrorIField
        OCTET STRING
}


--  Although the objects in this table are read-only, at the agent's discretion
--  they may be made read-write so that the management station, when
--  appropriately authorized, may be used to reset counters.

hippiIndex OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Unique sequential integer identifying the hippi media. This is the ifIndex
        value of the corresponding ifEntry."
    ::= { hippiEntry 1 }

hippiWordSize OBJECT-TYPE
    SYNTAX      INTEGER {
                    hippi32bitDataBus (1),
                    hippi64bitDataBus (2)
                }
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "This enumerated value indicates the size of the data bus associated with this
        particular hippi media. This defines whether the interface operates at 800 or
        1600 Mbits/s."
    ::= { hippiEntry 2 }

hippiEndPointType OBJECT-TYPE
    SYNTAX      INTEGER {
                    hippiSourceEndPoint (1),
                    hippiDestinationEndPoint (2)
```

```
                }
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "This enumerated value indicates whether this side of the hippi simplex
        connection is operating as a source or a destination."
    ::= { hippiEntry 3 }

hippiWordCountHigh OBJECT-TYPE
    SYNTAX      INTEGER (0..'7FFFFFFF'h)
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "The word counter represents the number of words transferred through the
        media. The counter is cleared at power up or reset. This is the portion of the
        count above 999,999,999. This object may be made read-write accessible to
        allow resetting the counter from an appropriately configured management
        station."
    ::= { hippiEntry 4 }

hippiWordCountLow OBJECT-TYPE
    SYNTAX      INTEGER (0..999999999)
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "The word counter represents the number of words transferred through the
        media. The counter is cleared at power up or reset. This is the portion of the
        count below 1,000,000,000.  This object may be made read-write accessible to
        allow resetting the counter from an appropriately configured management
        station."
    ::= { hippiEntry 5 }

hippiBurstCountHigh OBJECT-TYPE
    SYNTAX      INTEGER (0..'7FFFFFFF'h)
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Total number of bursts transferred by this media since power up or reset. This
```

```
        is the portion of the count above 999,999,999. This object may be made read-
        write accessible to allow resetting the counter from an appropriately
        configured management station."
    ::= { hippiEntry 6 }

hippiBurstCountLow OBJECT-TYPE
    SYNTAX      INTEGER (0..999999999)
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Total number of bursts transferred by this media since power up or reset. This
        is the portion of the count below 1,000,000,000. This object may be made
        read-write accessible to allow resetting the counter from an appropriately
        configured management station."
    ::= { hippiEntry 7 }

hippiPacketCountHigh OBJECT-TYPE
    SYNTAX      INTEGER (0..'7FFFFFFF'h)
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "The packet count represents the number of packets transferred on the media.
        The counter is cleared at power up or reset. This is the portion of the count
        above 999,999,999. This object may be made read-write accessible to allow
        resetting the counter from an appropriately configured management station."
    ::= { hippiEntry 8 }

hippiPacketCountLow OBJECT-TYPE
    SYNTAX      INTEGER (0..999999999)
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "The packet count represents the number of packets transferred on the media.
        The counter is cleared at power up or reset. This is the portion of the count
        below 1,000,000,000. This object may be made read-write accessible to allow
        resetting the counter from an appropriately configured management station."
    ::= { hippiEntry 9 }
```

```
hippiParityErrors OBJECT-TYPE
    SYNTAX      Counter
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "This counter represents the number of data parity errors that have occurred on
        the media since power up or reset. Since the parity bus is one directional from
        the source to the destination, this object only has meaning when the
        hippiEndPointType indicates this media is a destination end point. This object
        may be made read-write accessible to allow resetting the counter from an
        appropriately configured management station."
    ::= { hippiEntry 10 }

hippiOverrunCount OBJECT-TYPE
    SYNTAX      Counter
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Total number of overrun errors that have occurred on the media since power
        up or reset. An overrun is defined as the number of times the Lost status bit
        went active. Because of signal direction, this object only has meaning when the
        hippiEndPointType indicates this media is a destination end point. This object
        may be made read-write accessible to allow resetting the counter from an
        appropriately configured management station."
    ::= { hippiEntry 11 }

hippiInputPortRejectCount OBJECT-TYPE
    SYNTAX      Counter
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Total number of times that the input port has rejected a connection attempt
        due to I-Field parity errors, busy ports, or invalid source addresses since power
        up or reset. This object is only meaningful when the hippiEndPointType
        indicates this media is a destination end point. This object may be made read-
        write accessible to allow resetting the counter from an appropriately
        configured management station."
    ::= { hippiEntry 12 }
```

```
hippiCampOnDelayCount OBJECT-TYPE
    SYNTAX      Counter
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "This counter represents the number of times the port entered a camp on delay
        state while attempting to connect to a HIPPI destination since power up or
        reset. This object is only meaningful when the hippiEndPointType indicates
        this media is a source end point. Camp on is described in the HIPPI-SC
        specification [5]. Implementation of this counter is optional since it may not
        apply to all hippi media types. This object may be made read-write accessible
        to allow resetting the counter from an appropriately configured management
        station."
    ::= { hippiEntry 13 }

hippiDownlineRejectCount OBJECT-TYPE
    SYNTAX      Counter
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "This counter represents the number of connection requests that were rejected
        by a destination end point since power up or reset. This object may be made
        read-write accessible to allow resetting the counter from an appropriately
        configured management station."
    ::= { hippiEntry 14 }

hippiUplineRejectCount OBJECT-TYPE
    SYNTAX      Counter
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "This counter represents the number of times that the source end point dropped
        the connection request before the connection to the final destination was
        completed since power up or reset. This object may be made read-write
        accessible to allow resetting the counter from an appropriately configured
        management station."
    ::= { hippiEntry 15 }
```

```
hippiDownlineDisconnectCount OBJECT-TYPE
    SYNTAX      Counter
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "This counter represents the number of completed connections that were
        terminated by a destination end point since power up or reset. This object may
        be made read-write accessible to allow resetting the counter from an
        appropriately configured management station."
    ::= { hippiEntry 16 }

hippiUplineDisconnectCount OBJECT-TYPE
    SYNTAX      Counter
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "This counter represents the number of completed connections that were
        terminated normally by the source end point since power up or reset. This
        object may be made read-write accessible to allow resetting the counter from
        an appropriately configured management station."
    ::= { hippiEntry 17 }

hippiLastIField OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE (4) )
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "This data type specifies the HIPPI I-Field address that was associated with the
        last operation on the media. If the hippiEndPointType indicates this is a
        Destination then the I-Field gives an indication of how the end point was
        addressed. If the media is a Source then the I-Field indicates the last address
        that the media attempted to connect to."
    ::= { hippiEntry 18 }

hippiConnectState OBJECT-TYPE
    SYNTAX      INTEGER {
                    hippiDisconnect (1),
                    hippiConnectRequest (2),
```

```
                  hippiConnected (3)
              }
    ACCESS    read-only
    STATUS    mandatory
    DESCRIPTION
        "Indicates the current connection state of the media."
    ::= { hippiEntry 19 }

hippiLastErrorType OBJECT-TYPE
    SYNTAX    INTEGER {
                  hippiErrorTypeReject (1),
                  hippiErrorTypeBusy (2),
                  hippiErrorTypeCampOnTimeout (3),
                  hippiErrorTypeParityError (4),
                  hippiErrorTypeInvalidAddress(5)
              }
    ACCESS    read-only
    STATUS    mandatory
    DESCRIPTION
        "Contains the type identifier of the last error that was encountered on this
        media. The type identifier and the hippiLastErrorIField values are intended to
        be used together to aid in determining line connection problems."
    ::= { hippiEntry 20 }

hippiLastErrorIField OBJECT-TYPE
    SYNTAX    OCTET STRING (SIZE (4) )
    ACCESS    read-only
    STATUS    mandatory
    DESCRIPTION
        "This data type specifies the HIPPI I-Field address that was associated with the
        last error for this media. The hippiLastErrorType along with this I-Field
        information is intended to help in determining connection problems. If the
        hippiEndPointType indicates this is a Destination then the I-Field gives an
        indication of how the end point was addressed. If the media is a Source then
        the I-Field indicates the last address that the media attempted to connect to"
    ::= { hippiEntry 21 }

END
```

7. Glossary

There are a number of terms used in this document that are specific to discussions regarding HIPPI. For convenience, this glossary contains definitions from the HIPPI-PH [4] and HIPPI-SC [5] for those terms.

burst. A group of 1 to 256 words. Bursts that contain less the 256 words are called short bursts. Short bursts in a packet indicate either the first or last burst of the packet.

connection. Condition of the HIPPI-PH when data transfers from a Source end-point to a Destination end-point are possible.

destination. The equipment at the end of the interface that receives the data.

end-point. The equipment at either end of the fabric for a particular connection.

I-Field. A 32-bit information field sent as part of the sequence of operations establishing a connection from a Source to a Destination.

logical address. An address stored in an I-Field that uniquely identifies a Destination end-point or set of end-points.

packet. A data set, as defined by HIPPI-PH, sent from Source to Destination. A packet is composed of one or more bursts.

source. The equipment at the end of the interface that transmits the data.

source address. An address stored in an I-field that uniquely identifies a Source end-point or set of end-points.

word. A unit of information consisting of 32 or 64 bits, matching the DATA BUS width, and transferred from the Source to the Destination.

8. Acknowledgments

The following people provided additional comments and suggestions:

Bob Kowalski (bjk@network.com)

Jim Hughes (hughes@network.com)

9. References

[1] 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.

[2] McCloghrie K., and M. Rose, Editors, “Management Information Base for Network Management of TCP/IP-based internets”, RFC 1213, Performance Systems International, March 1991.

[3] Rose, M., Editor, “A Convention for Defining Traps for use with the SNMP”, RFC 1215, Performance Systems International, March 1991.

[4] High Performance Parallel Interface, Mechanical, Electrical, and Signalling Protocol Specification (HIPPI-PH), ANSI X3.183-1991

[5] High Performance Parallel Interface, Switch Control (HIPPI-SC), ANSI Draft X3.222-199x. Global Engineering, Irvine, CA.

10. Security Considerations

Security issues are not discussed in this memo.

11. Author’s Address

Jeff Miller
Network System Corporation
7600 Boone Avenue North
Minneapolis, Minnesota 55428

Phone: (612) 424-4888

EMail: Jeff.Miller@network.com