New Technical Notes

Macintosh ®

Developer Support

NW 19 - How the Simple Network Management Protocol (SNMP) Manager Finds Network Cards

Networking

Written by: Mike Ritter June 1993

Apple provides an SNMP Manager that implements an architecture for doing network management on a Macintosh computer. This Tech Note describes how the SNMP manager finds network cards on a Macintosh and explains how you can make the SNMP Manager recognize new types of network cards. This Tech Note is a supplement to the information provided in the Developers kit for SNMP available on the E.T.O. CD-ROM in the *MacSNMP Programmer's Guide.*. The kit defines the interfaces to actually gather the information.

Topics

- How the SNMP Manager finds a card's ifGroup statistics
- How the SNMP Manager finds a card's link layer statistics
- How to add a new card type that the SNMP Manager will understand

Introduction

This Tech Note assumes that you know something about SNMP and how it works. It also assumes that you know something about the Macintosh computer's resource manager and driver architecture. This note provides information additional to that provided in the *MacSNMP Programmer's Guide*. The method that the SNMP Manager uses to find a networking card depends on the slot manager; therefore, machines that do not support the slot manager will not support SNMP-managed network cards.

Network Management Framework

In the IETF (Internet Engineering Task Force) framework for network management there are MIBs (Management Information Bases) for practically every known device, including Ethernet cards, Token Ring cards, FDDI cards, LocalTalk, SMDS, ATM, and others. A MIB describes a minidatabase that a device must support in order to be network manageable. MIBs consist of groups, each of which contains variables and/or rows of variables (tables). One of the standard groups in the standard TCP/IP MIB II (RFC 1213) is the interfaces (if) group. The ifGroup contains variables that describe any (and *every*) network card. It defines statistics (identified as variables) such as number of packets sent, number of bytes sent, number of bytes received, and so on. As documented in the *MacSNMP Programmer's Guide,* the SNMP Manager makes a driver call to a network card to determine this information. But how does it find the driver?

Finding the Driver

The SNMP Manager looks through all of the slots (including the pseudo slot for a `.ENET0` device) using the Slot Manager. It searches for any device that has the `SpBlock.spCategory` set to `catNetwork` (=4.) It assumes that this (and *only* this category) is a network card. It then looks in its `STR#` resource (resource ID = 1) for the driver name, using the `SpBlock.spCType` as the index. The shipping resource is listed below.

```
resource 'STR#' (1,"Driver Names for cards"){
{
".ENET",".TOKN","","","","","","",
"","","","","","","","",
".FDDI"
}};
```

Please note that since the `spCType` of an FDDI card is 17, there are 14 blank entries in this resource. When the original (pre-FDDI final specifications) code for the SNMP Manager was done, FDDI was going to have an `spCType` of 3. It is unknown why this changed, but Apple Developer Technical Service has promised to be more compact in the future.

The reason for this hack is that the driver name for many cards is not correctly stored in the `SpBlock`, while the `spCType` always is. The SNMP Manager then checks to see if a driver of that name is opened for that slot. If it is, it can then proceed to make the parameter block based statistics calls to gather the information defined in the ifTable in the ifGroup.

The ifTable in the ifGroup is defined in ASN.1 notation in RFC 1213 as:

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

The SNMP Manager controls the index, as this must be the same for various other tables. Additionally, the ifTable requires that we return an object ID that describes the `ifSpecific` system on the card and an integer describing the `ifType`. This tells network management stations where to go to get information specific to this type of network card (things like how many times a Token Ring got a bad token or how many times packet length overruns occurred on an Ethernet card). The SNMP Manager finds this `object identifier` by using the `SpBlock.spCType` as the index into another resource of type 'ifSP'. It also finds the ifType in this resource; this number is defined in RFC 1213 for several networking systems. This resources is defined as:

```
/*-------------ifSP • ifSpecific Enumerated Object IDs-----------------*/
type 'ifSP' {
   integer = $$Countof(OIDEnum);       /* Number of specific Object IDs */
   wide array OIDEnum {
          unsigned long;                     /* ifType, from RFC 1213 */
          wide array ObjectID {              /* Object ID for this card type */
                 unsigned integer = $$Countof(ID);   /* Num of IDs in OID */
                 wide array ID {
                        unsigned long;                     /* an ID */
                                    };
                 };
          };
};
```

And the resource that is shipping with the card looks like the following:

```
 data 'ifSP' (1, "ifSpecific Enumerated") {
     $"0011"             //Three entries in table
     $"0000 0006"   // Enet type
     $"0008"             //ENET Object ID
     $"0000 0001"
     $"0000 0003"
     $"0000 0006"
     $"0000 0001"
     $"0000 0002"
     $"0000 0001"
     $"0000 000A"
     $"0000 0007"

     $"0000 0009"   // Token ring type
     $"0008"             //Token ring Object ID
     $"0000 0001"
     $"0000 0003"
     $"0000 0006"
     $"0000 0001"
     $"0000 0002"
     $"0000 0001"
     $"0000 000A"
     $"0000 0009"

     $"0000 0000 0000"      // #3  type + oid ID
     $"0000 0000 0000"      // #4  type + oid ID
     $"0000 0000 0000"      // #5  type + oid ID
     $"0000 0000 0000"      // #6  type + oid ID
     $"0000 0000 0000"      // #7  type + oid ID
     $"0000 0000 0000"      // #8  type + oid ID
     $"0000 0000 0000"      // #9  type + oid ID
```

```
        $"0000 0000 0000"      // #10 type + oid ID
        $"0000 0000 0000"      // #11 type + oid ID
        $"0000 0000 0000"      // #12 type + oid ID
        $"0000 0000 0000"      // #13 type + oid ID
        $"0000 0000 0000"      // #14 type + oid ID
        $"0000 0000 0000"      // #15 type + oid ID
        $"0000 0000 0000"      // #16 type + oid ID

        $"0000 000F"   // FDDI Type
        $"0008"             //FDDI Object ID
        $"0000 0001"
        $"0000 0003"
        $"0000 0006"
        $"0000 0001"
        $"0000 0002"
        $"0000 0001"
        $"0000 000A"
        $"0000 000E"
    };
```

Please note again that since the `spCType` of FDDI is 17, there are 14 blank entries in this resource.

Both of these resources are stored in the SNMP Manager file in the Extensions folder. If you are releasing a new type of networking card and want it to be network manageable, you must perform the following steps.

To support the generic ifGroup (minimum support to claim network compatibility) you must

- implement the driver call described in the *MacSNMP Programmer's Guide* for network cards; specifically, it must support the following driver selector: 242 (`LapGetLinkStatus`).
- ensure that the STR# resource of ID #1 in the SNMP Manager file in the Extensions folder has the name of your network driver in the correct string (the one that has an index equal to the `spCType` of your card).
- ensure that the 'ifSP' resource of ID #1 in the same place has the correct entry for your network card's object ID. This object ID will only be used for display.

When adding your entry to either of these resources, please don't overwrite the entire resource. Just add your specific entries to the resource. Other cards or future release may have already altered the resources beyond what is listed above and you would not want to erase useful information and cause other people's hardware to fail.

To support the specific MIB for your network group is a bit more complicated. The shipping SNMP Manager defines a driver interface for Ethernet and Token Ring cards to use to return the network specific statistics. And future versions of the AppleTalk Connection Product will have Agents that can actually report these values to the SNMP Manager. The Agent will depend upon the correct driver name from the STR# resource described above. Thus, if you are building an Ethernet or Token Ring card you will only have to implement the correct driver calls as documented in the *MacSNMP Programmer's Guide* to become fully network manageable.

If you are building a third type of card, the problem is more difficult. First the IETF has not defined MIBs for most other types of cards, and second there is no definition for which

selector returns which information from the card's driver. As new cards become available Apple may provide Link Layer agents that can manage these new cards. Apple Developer Technical Support should be notified if you are building a new card. There are several procedures to define new card interfaces, and they can recommend the best alternative for your card type.

Conclusion

Please talk to Apple Developer Technical Support if you are developing a new type of networking card for the Macintosh.

Further Reference:

- *Inside Macintosh*, Volume V, Slot manager
- *Designing Cards and Drivers for the Macintosh Family*
- *MacSNMP Programmer's Guide*
- *RFC 1157, A Simple Network Management Protocol*
- *RFC 1213, Management Information Base for Network Management of TCP/IP-Based Internets: MIB-II*
- *RFC 1155, Structure and Identification of Management Information for TCP/IP- Based Internets*