Getting Started with MRV Communications LX Series MIBs

Corporate Headquarters

MRV Communications, Inc. Corporate Center
20415 Nordhoff Street
Chatsworth, CA 91311
Tel: 818-773-0900
Fax: 818-773-0906
www.mrv.com (Internet)

Sales and Customer Support

MRV Americas
295 Foster Street
Littleton, MA 01460
Tel: 800-338-5316 (U.S.)
Tel: +011 978-952-4888
(Outside U.S.)
sales@mrv.com (email)
www.mrv.com (Internet)

MRV International
Industrial Zone
P.O. Box 614
Yokneam, Israel 20682
Tel: 972-4-993-6200
sales@mrv.com (email)
www.mrv.com (Internet)

451-0314C

Contents

Introduction

This guide provides end-users of MRV Communications' LX units with basic information regarding the Network Management System (NMS), and procedures on how to use the Management Information Base (MIB) structure (as pointers to objects in the devices) to manage these units. This guide also provides the location of MRV Communications' Proprietary and Standard MIBs, how they can be obtained, as well as instructions on downloading and compiling them to wherever their application specifies.

Network Management System

The following details the Network Management System and how the Management Information Base (MIB) is used with network management protocols in TCP/IP-based Internets.

Network Management Systems execute management applications that monitor and control network elements. Network Elements (NE) are devices such as hosts, routers, terminal servers, etc., that are monitored and controlled through access to their management information.

The Network Management System can potentially monitor several nodes, each with a processing entity termed an agent. An agent is a network management software module that resides in a managed device. It has local knowledge of management information and can translate that information into a form compatible with SNMP. Agents are entities that interface to the actual device being managed. These managed objects might be hardware, configuration parameters, performance statistics, and so on, directly relating to the current operation of the device in question. The agent has access to at least one management station and a management protocol used to convey management information between the agents and management stations. Operations of the protocol are carried out under an administrative framework that defines both primitive authentication and authorization policies in SNMPv1, SNMPv2C, and SNMPv3.

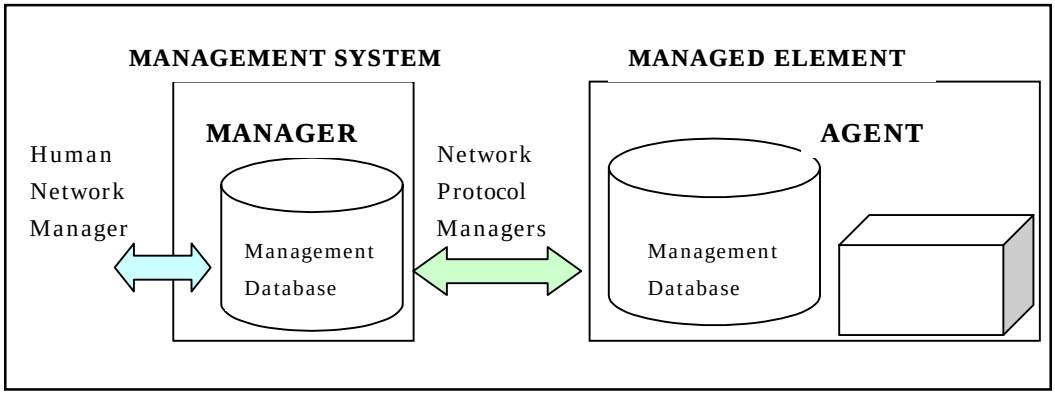

Figure 1 - Typical Network Management System

All SNMP managed devices contain a Management Information Base (MIB) database that stores management information for that device pertinent to network management. A MIB is a collection of information organized hierarchically.

The database is organized as a tree; branches of the tree name objects and the leaves of the tree contain the values manipulated to effect management. The values are comprised of managed objects and are identified by object identifiers. Objects in the MIB are defined using Abstract Syntax Notation One (ASN.1). The concepts of this tree are called out in STD 16/RFC 1155, “The Structure of Management Information” or SMI. The SMI defines the trunk of the tree and the types of objects used when defining the leaves. STD 16/RFC 1212, “Towards Concise MIB Definitions”, defines a more concise description mechanism that preserves all the principles of the SMI.

A managed object, (sometimes called a MIB object, an object, or a MIB) is one of any number of characteristics of a managed device. Managed objects are comprised of one or more object instances, which are essentially variables. Each managed device has a unique address. Furthermore, each managed object per managed device also has a unique address. These unique addresses are known as Object Identifiers (OID). Each enterprise (company) subscribing to the SNMP System is provided with a unique OID, and the enterprise in turn, will allocate unique OIDs to each of its managed objects. OIDs are contained within SNMPs Management Information Base (MIB), which is a virtual blueprint of OIDs serving as the common dictionary for SNMP communications.

Names are used to identify managed objects that use the Object Identifier concept to model this notation. An Object Identifier is a sequence of integers that traverse a global tree. The tree consists of a root connected to a number of labeled nodes via edges. Each node may, in turn, have children of its own which are labeled. In this case, we may term the node a subtree. This process may continue to an arbitrary level of depth. Central to the notion of the Object Identifier is the understanding that administrative control of the meanings assigned to the nodes may be delegated as one traverses the tree. A label is a pairing of a brief textual description and an integer.

Example of an OID Structure:

Internet OBJECT IDENTIFIER ::= {iso (1) org (3) dod (6) internet (1) 1}

Internet=Name

1.3.6.1 (iso.org.dod.internet) =Object Identifier (OID)

If read from the hierarchal tree structure, it would appear as follows:

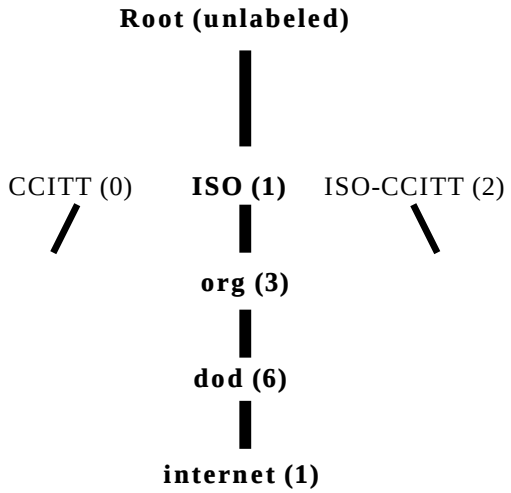

Figure 2 - Hierarchical Tree Structure

Core MIB definitions for the Internet suite of protocols can be found in RFC 1155, Management Information Base for Network Management of TCP/IP-based Internets. STD 17/RFC 1213 defines MIB-II, an evolution of MIB-I with changes to incorporate implementation experience and new operational requirements. STD 15/RFC 1157 defines the SNMP protocol itself. The protocol defines how to manipulate the objects in a remote MIB.

NOTES:

1. The Simple Network Management Protocol (SNMP) is an Internet standard defined by the Internet Engineering Task Force (IETF) Request for Comment (RFC) 1157, which specifies how network management information is carried through a network.
2. MRV Communications' devices support SNMP by implementing an SNMP Agent. The agent stores SNMP Management Information Base data and makes it available when requested via SNMP Get/Set requests.
3. In addition, these devices generate SNMP Trap messages, which are indications that specific events have occurred.

The definition of an object in the MIB requires an object name and type. Object names and types are defined using the subset of Abstract Syntax Notation One (ASN.1), as defined in the SMI. Objects are named using object identifiers, administratively assigned names to specify object types. The object name, together with an optional object instance, uniquely identifies a specific instance of an object. A textual convention string, termed the OBJECT DESCRIPTOR, may be used to identify the object.

Textual conventions enhance the readability of the specification and can ease comparison with other specifications if appropriate. It should be noted that the introduction of textual conventions has no effect on either the syntax or the semantics of any managed objects.

These conventions are merely an artifact of the explanatory method used. Objects defined in terms of one of these methods are always encoded by the rules that define the primitive type. Hence, no changes to the SMI or the SNMP are necessary to accommodate these textual conventions, which are adopted for the convenience of readers and writers in pursuit of the elusive goal of clear, concise and unambiguous MIB documents. For example, an ASCII “DisplayString” is a textual convention defined in RFC 1213, and is built on ASN.1 data type “OctetString”, but with added refinement specific to ASCII display strings.

Objects have a syntax that defines the abstract data structure corresponding to the object type. The ASN.1 language provides the primitives used for this purpose. The SMI purposely restricts the ASN.1 constructs, which may be used for simplicity and ease of implementation. The encoding of an object type, simply describes how to represent an object using ASN.1 encoding rules for purposes of dealing with the SNMP protocol.

Management information is a collection of managed objects, residing in a virtual information store called the Management Information Base. Collections of related objects are defined in MIB modules and are written using a subset of ASN.1. The subset is defined by the SMI and is divided into three parts:

1. Module definitions are used when describing information modules. An ASN.1 macro MODULE-IDENTITY is used to convey the semantics of an information module.
2. Object definitions are used when describing managed objects. An ASN.1 macro OBJECT-TYPE is used to convey the syntax and semantics of a managed object.
3. Notification definitions are used when describing unsolicited transmissions of management information. An ASN.1 macro TRAP-TYPE is used to convey the syntax and semantics of a trap.

MIBs are organized into MIB modules. A MIB module is a file defining all the MIB objects under a subtree. The foundation module is the standards-based MIB-II module defined by RFC 1213. *(In addition to the Internet-standard MIB-II objects defined in RFC 1213, hardware vendors, such as MRV Communications, Hewlett-Packard, and Cisco Systems have developed MIB extensions for their own products. A MIB defined by a specific vendor is referred to as an enterprise-specific MIB).* See the “MIB Classifications” section for a list of MRV enterprise-specific MIBs.

As mentioned earlier, MIB objects are organized in a hierarchical tree structure. The root node itself is unlabeled, but has at least three children directly under it. One node is administered by the International Organization for Standards, with label ISO (1); another is administered by International Telegraph and Telephone Consultative Committee, with label CCITT (0); and the third is jointly administered by ISO and CCITT, Joint – ISO – CCITT (2) (see Figure 2). Each branch in the tree has a unique name and numeric identifier. Intermediate branches of the tree serve as a way to group related MIB objects together.

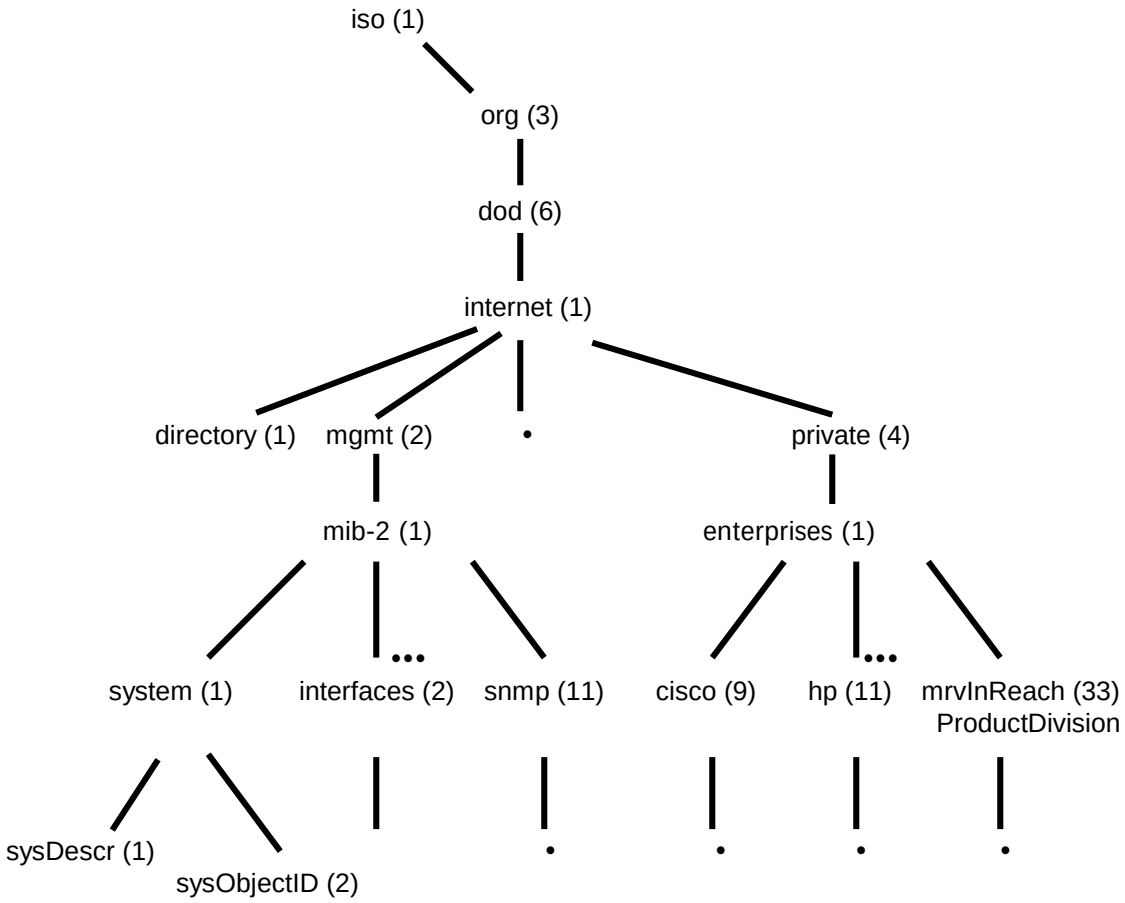

Figure 3. MIB Organization

The “leaves” of the tree represent the actual MIB object. A subtree refers to the entire group of branches and leaves under a particular intermediate branch. Figure 1-3 illustrates the tree and subtree structures.

A MIB object is uniquely identified by its place in the tree. A full object identifier consists of the identifier of each branch along the path through the tree hierarchy, from the top of the tree **“iso”**, down to the leaf **“sysObjectID”** as illustrated in Figure 3. The object identifier is expressed in “dotted notation”, by separating each branch identifier along the path with a period.

The **"mib-2"** subtree is **iso.org.dod.internet.mgmt.mib-2** and its numeric identifier is **1.3.6.1.2.1**. As another example, the full MIB object identifier for **"sysObjectID"** is **iso.org.dod.internet.mgmt.mib-2.system.sysObjectID** and its numeric identifier is **1.3.6.1.2.1.1.2**.

The instance identifier on a MIB object with only one instance is zero. The instance identifier on a MIB object with more than one instance is one or greater. MIB object notations follow the standard notation defined in ASN.1. The ASN.1 standard notation definition can be considered the 'template' for MIBs.

To avoid conflicts of object IDs, each branch of the tree must be registered, that is, defined through a designated organization. For example, the Internet Activities Board (IAB), has authority over the **internet** subtree, which includes the MIB-II Internet standard registered under the **"mib-2"** subtree. In turn, the IAB gives vendors authority over enterprise-specific subtrees. Enterprise-specific MIB objects are registered under the designated authority for that enterprise. To clarify this point, MRV Communications would register its enterprise-specific MIBs under **1.3.6.1.4.1.33**, having authority over the **enterprises.mrv** subtree.

The **sysObjectID** is an important MIB object to management platforms, such as, HP OpenView. The **sysObjectID** is registered in the Internet-standard MIB-II module as **iso.org.dod.internet.mgmt.mib-2.system.sysObjectID (1.3.6.1.2.1.1.2)**. The **sysObjectID** is used for administrative purposes to uniquely identify the type of agent software that is running on a given vendor's hardware. This object is different from most other MIB objects. When queried, this object sends back an object identifier that describes the product.

Example of SNMPGet for SysObjectID on LX:

NOTE: The exact syntax of the SNMP Get request depends on the management platform.

```
C:\SNMP>snmpget 140.179.xxx.xxx 1.3.6.1.2.1.1.2.0
SNMP++ GET to 140.179.xxx.xxx SNMPV1 Retries=1 Timeout=100m Community=Public
Oid = 1.3.6.1.2.1.1.2.0
Value = 1.3.6.1.4.1.33.8.1.31
```

MRV Communications - MIB Modules

LX units support the **lx-subscriber-mib** proprietary mib, which is listed in Appendix A.

Security

By default, MRV Communications' network devices accept SNMP GET and SET requests from the Network Operations Center (NOC). SNMP access can be restricted to the device by specifying the SNMP Clients and Communities. A Community refers to one or more NOCs that specify the same Community string in their SNMP messages. A Client is a specific NOC, which is identified through an IP or Ethernet Address. This can be accomplished by the Command Line Interface (CLI) or via any SNMP package using the SET command.

Managing SNMP Clients and Communities in the LX CLI

This section describes how to define SNMP Clients and Communities, set miscellaneous SNMP values, and display SNMP-related information.

The tasks in this section are performed in the LX Command Line Interface (CLI). Refer to the *LX-Series Commands Reference Guide* (451-0310) for more information on the commands that are used in this section.

Defining a Trap Client

Execute this command at the SNMP command mode. An LX will not generate an SNMP Trap message until a Trap Client is defined. A Trap Client is a specific NOC to which the Element Manager sends Trap messages. One of more Trap Clients can be defined through this command:

`Snmp:0 >>trap client` *`number ip-address`*

A *number* value is a number from 0 to 15. The *ip-address* identifies the NOC that should receive the Trap messages. For example:

```
Snmp:0 >>trap client 1 140.179.12.3
```

SNMP SET Example

If a new trap client is to be defined, add it to the trap client table as follows:

1. Walk the SNMP Trap Client Table looking for a client number with an address of zero. An example is `SNMP NEXT ObjectID: 1.3.6.1.4.1.33.10.3.9.1.4`.
2. Add the new address using `SNMP SET`. The exact command syntax will vary depending on the SNMP application in use. An example is `SNMP SET objectID: 1.3.6.1.4.1.33.10.3.9.1.4.3 Type: OctetString value: 140.179.1.1.`

Defining Get and Set SNMP Clients

Execute this command at the SNMP command mode. A GET Client is a specific NOC that is allowed to manage the In-Reach Element Manager through GET and GET_Next requests. A SET Client is a NOC that may issue SET Requests to the Element Manager. You can use the following commands to define up to four of each of these client types:

`Snmp:0 >>get client [`*`number`*`]` *`ip-address`*
`Snmp:0 >>set client [`*`number`*`]` *`ip-address`*

A *number* value is a number from 0 to 15. Define a previously defined Get or Set client to 0.0.0.0 in order to remove it.

Example

```
Snmp:0 >>set client 1 123.223.123.1
```

SNMP SET Example

```
Set client: SET 1.3.6.1.4.1.33.50.1.2.1.6 Integer 6
Get client: GET 1.3.6.1.4.1.33.50.1.2.1.6 Integer 6
```

Defining SNMP Communities

Execute the `get/set/trap client` command at the SNMP command mode. GET and SET Communities provide an additional level of security. If you do not define any GET Clients, the LX unit will accept GET and GET_Next requests from any NOC whose GET Requests include a Community name that matches the LX unit's GET Community. If you do not define a GET Community, the LX unit will accept GET and GET_Next Requests from any NOC. Similarly, if you do not define any SET Clients, the LX unit will accept SET Requests from any NOC whose requests include a Community name that matches the LX unit's SET Community. If you do not define a SET Community, the LX unit will accept SET Requests from any NOC.

If a Trap Community is defined, the LX unit will include the Trap Community name in the Trap messages that it generates. Use the following commands to define GET, SET, and Trap Community names:

`Snmp:0 >>get client [`*`number`*`] `*`community word`*

A *community* can include up to 32 characters.

`Snmp:0 >>set client [`*`number`*`] `*`community word`*

`Snmp:0 >>trap client [`*`number`*`] `*`community word`*

Examples

```
Snmp:0 >>get client 1 community none
```

```
Snmp:0 >>set client 2 community in-reach
```

Miscellaneous SNMP Settings

This section explains how to define SNMP Contact, Name, and Location strings.

SNMP Contact

Execute this command at the SNMP command mode. An SNMP Contact, or sysContact, identifies a person to contact when the LX unit needs attention. Use this command to define a contact:

`Snmp:0 >>contact `*`"contact-strin`*`g"`

The "*contact-strin*g" can include up to 60 characters, e.g., "John Smith, 800-555-1212"

Example

```
Snmp:0 >>contact bobby_jones
```

Displaying the SNMP Characteristics

Use the following command to display the system-wide SNMP characteristics for the LX unit:

```
In-Reach:0 >>show snmp characteristics
```

Example

```
In-Reach:0 >>show snmp characteristics
```

```
Time:   Wed, 10 Apr 2002 10:45:08 UTC        Name:          In-Reach
Logging:          Disabled                   Port:              161
Contact:       Henry Smith                   Location:  Upstairs Lab
V3 Engine Boots:         1
V3 Engine ID:            6537303330653865313136323936336100000000
```

SNMP CHARACTERISTICS Display

Displaying the SNMP Client

Use the following command to display the Community status (public or private) and the Get, Set, and Trap versions of an SNMP client:

```
In-Reach:0 >>show snmp client number
```

A *number* value is any valid client number from 0 to 16.

Example

```
In-Reach:0 >>show snmp client 1
```

```
Client:                  2
 Get Client:         0.0.0.0          Get Community:       public
 Get Version:              1
 Set Client:         0.0.0.0          Set Community        public
 Set Version:              1
 Trap Client:        0.0.0.0          Trap Community       private
 Trap Version:             1
```

IP SNMP Client Display

Displaying the SNMP V3 Settings

Use the following command to display the V3 settings for a Version-3 SNMP client:

```
In-Reach:0 >>show snmp v3 number
```

A *number* value is any valid SNMP V3 client number from 0 to 16.

Example

```
In-Reach:0 >>show snmp v3 1
```

```
Client:                          4
Engine Boots:                    8
Engine ID:                        e70303eeace2923a00000000000000000000000000
V3 View OID Subtree:             0
V3 View OID Subtree Mask:         800000000000000000000000000000000000000000
V3 View Name:
V3 View Type:                 Allow
V3 Access Group Name:                V3 Access Context Prefix:
V3 Access Security Model:        3   V3 Access Security Level:          1
V3 Access Read View:           view   V3 Access Write View:           view
V3 Access Notify View:         view
V3 Access Cont. Match:         none
V3 Security Name:                    V3 Source IP:               0.0.0.0
V3 Secur. Community:         public   V3 Source Mask:             0.0.0.0
V3 Group Secur. Model:               V3 Group Secur. Name:
V3 Group Name:
```

Compiling MIBs

It may be a good practice to open the MIB prior to compiling it. Use the mib compiler associated with your NMS to compile the MIBs. For HP Openview, as the user clicks on the MIB file to add it to the database, the compiler checks it for errors. When individual MIBs are being added to the database, the order in which the MIBs are added is important, since one MIB often references objects in other MIBs.

Some compilation errors can be fixed by commenting out a line or an entire objected with double-hyphens, or "—" at the beginning of each unwanted line. Remove any (comment indications) "—" later in that line. The user can do this if not interested in managing that particular object.

Appendix A

LX MIBs

This MIB implements the objects related to the LX subscriber (or user) configuration and status.

LX Subscriber MIB

```
-- $Revision: 1.9 $

IN-REACH-Subscriber-mib DEFINITIONS ::= BEGIN

--
--
-- Subscriber MIB Text File
--
-- Date: Thurs. Feb 4, 2003 user: DA
-- File created using EMACS
-- By: DA
--
-- FILE: lx-subscriber-mib.mib
--
        -- import modules

           IMPORTS
             Counter, Gauge, TimeTicks, IpAddress
                    FROM RFC1155-SMI
             DisplayString
                    FROM RFC1213-MIB
             OBJECT-TYPE
                    FROM RFC-1212;
        --     mrvInReachProductDivision
        --            FROM MRV-IN-REACH-PRODUCT-DIVISION-MIB;
        --
        --     DisplayString, sysLocation
        --            FROM RFC1213-MIB
        --     TRAP-TYPE
        --            FROM RFC-1215
        --     iTouch, DateTime, AddressType
        --            FROM ITOUCH-MIB
        --     charPortIndex
        --            FROM RFC1316-MIB
        --     rs232InSigState, rs232OutSigState, rs232PortIndex
        --            FROM RFC1317-MIB;
        --
        -- definition of object types
```

```
--
--  OBJECT-TYPE MACRO ::=
--    BEGIN
--      TYPE NOTATION ::= "SYNTAX" type (TYPE ObjectSyntax)
--                        "ACCESS" Access
--                        "STATUS" Status
--      VALUE NOTATION ::= value (VALUE ObjectName)
--
--      Access ::= "read-only"
--                       | "read-write"
--                       | "write-only"
--                       | "not-accessible"
--      Status ::= "mandatory"
--                       | "optional"
--                       | "obsolete"
--  END
--
-- names of objects in the MIB
--
--     ObjectName ::= OBJECT IDENTIFIER
--
-- syntax of objects in the MIB
--
--    ObjectSyntax ::=
--           CHOICE {
--               simple
--                   SimpleSyntax,
--
       -- note that simple SEQUENCEs are not directly
       -- mentioned here to keep things simple (i.e.,
       -- prevent mis-use).  However, application-wide
       -- types which are IMPLICITly encoded simple
       -- SEQUENCEs may appear in the following CHOICE
--
--               application-wide
--                   ApplicationSyntax
--           }
--
--          SimpleSyntax ::=
--              CHOICE {
--                  number INTEGER,
--                  string STRING,
--                  object OBJECT IDENTIFIER,
--                  empty  NULL
--              }
--
--          ApplicationSyntax ::=
```

```
--              CHOICE {
--                  address NetworkAddress,
--                  counter Counter,
--                  gauge Gauge,
--                  ticks TimeTicks,
--                  arbitrary Opaque
--
          -- other application-wide types, as they are
          -- defined, will be added here
--
--              }
--
--
          -- application-wide types
--
--          Counter ::=
--              [APPLICATION 1]
--                  IMPLICIT INTEGER (0..4294967295)
--
--          Gauge ::=
--              [APPLICATION 2]
--                  IMPLICIT INTEGER (0..4294967295)
--
--          TimeTicks ::=
--              [APPLICATION 3]
--                  IMPLICIT INTEGER (0..4294967295)
--
--          Opaque ::=
--              [APPLICATION 4]
--                  IMPLICIT OCTET STRING
--
-- Define OIDs

-- ccitt        OBJECT IDENTIFIER ::= { 0 }
-- null         OBJECT IDENTIFIER ::= { ccitt 0 }

-- iso          OBJECT IDENTIFIER ::= { 1 }
-- org          OBJECT IDENTIFIER ::= { iso 3 }
-- dod          OBJECT IDENTIFIER ::= { org 6 }
-- internet     OBJECT IDENTIFIER ::= { dod 1 }
-- directory    OBJECT IDENTIFIER ::= { internet 1 }
-- mgmt         OBJECT IDENTIFIER ::= { internet 2 }
-- experimental OBJECT IDENTIFIER ::= { internet 3 }
-- private      OBJECT IDENTIFIER ::= { internet 4 }
-- enterprises  OBJECT IDENTIFIER ::= { private 1 }
```

```
          mrvInReachProductDivision OBJECT IDENTIFIER ::= {enterprises
33}

          lxagent        OBJECT IDENTIFIER ::=
{mrvInReachProductDivision 50}

         --
             -- In-Reach         OBJECT IDENTIFIER ::= { enterprises 33
}
             -- agent           OBJECT IDENTIFIER ::= { In-Reach 8 }
             --
             -- A In-Reach agent identifer has the following fields:
             --
             --     In-Reach.agent.software.variant.version
             --
             -- Where:
             --
             --     In-Reach.agent   is an ordinary OID prefix.
             --                      Note that In-Reach's original form
for
             --                      such OID's used the value 1 for agent
             --                      and had no variant or version.  Later
             --                      forms included variant and version in
             --                      an inconsistent manner.
             --
             --     software         a value of the In-Reach SoftwareType
             --                      textual convention.
             --
             --     variant          a variant within a software type,
             --                      typically hardware dependent.  If the
             --                      software has no variants, it uses the
             --                      value 1.
             --
             --     version          a version number within the variant,
             --                      increased by one each time that
             --                      variant's agent changes in a way that
             --                      is significant to its MIB
             --                      capabilities description
             --
             -- Terminal Servers

              terminalServer   OBJECT IDENTIFIER ::= { lxagent 1 }

             -- This is a MIB module for all IN-REACH LX systems.
             --
             -- This MIB document is supplied "AS IS," and IN-REACH
             -- makes no warranty, either express or
```

```
  -- implied, as to the use operation, condition, or
  -- performance of the MIB.
  --
  -- Textual Conventions

sizeOfSubscriberTable OBJECT-TYPE
    SYNTAX  INTEGER (1..65534)
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
    "The size of the subscriberTable."
    ::= { terminalServer 1 }

subscriberTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF SubsEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
    "A list (table) of subscriber entries."
    ::= { terminalServer 2 }

subsEntry OBJECT-TYPE
    SYNTAX  SubsEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
    "A subscriber entry containing objects for a particular
    subscriber."
    INDEX   { index }
    ::= { subscriberTable 1 }

SubsEntry ::=
    SEQUENCE {
        index
            INTEGER,
        name
            DisplayString,
        usePassword
            INTEGER,
        password
            DisplayString,
        securityLevel
            INTEGER,
        maxConnections
            INTEGER,
        currentConnections
            Gauge,
```

```
                    activeUserIndex
                        INTEGER,
                    portSecurityList
                        DisplayString,
                    telnetAccess
                        INTEGER,
                    sshAccess
                        INTEGER,
                    guiAccess
                        INTEGER,
                    consoleAccess
                        INTEGER,
                    dialback
                        INTEGER,
                    dialbackNumber
                        DisplayString,
                    dialbackRetry
                        INTEGER,
                    useMenu
                        INTEGER,
                    menuName
                        DisplayString,
                    prompt
                        DisplayString,
                    termType
                        DisplayString,
--                     sessionMode
--                         INTEGER,
                    idleTimeout
                        INTEGER,
                    sessionTimeout
                        INTEGER,
                    localSwitch
                        DisplayString,
                    forwardSwitch
                        DisplayString,
                    backwardSwitch
                        DisplayString,
                    pause
                        INTEGER,
                    debugging
                        INTEGER,
                    logging
                        INTEGER,
                    preferredService
                        DisplayString,
                    dedicatedService
```

```
                    DisplayString,
                telnetLineMode
                    INTEGER,
                telnetEscapeChar
                    OCTET STRING,
                telnetSendcrlf
                    INTEGER,
                telnetReceivecrlf
                    INTEGER,
                sshCipher
                    INTEGER,
                remoteSshName
                    DisplayString,
                sshPort
                    INTEGER,
                sshLogLevel
                    INTEGER
            }
index OBJECT-TYPE
            SYNTAX  INTEGER (1..65534)
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
            "A unique index value for each subscriber."
            ::= { subsEntry 1 }

name OBJECT-TYPE
            SYNTAX  DisplayString (SIZE (0..80))
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
            "The Name of the subscriber entry."

            ::= { subsEntry 2 }

usePassword OBJECT-TYPE
            SYNTAX  INTEGER { disabled(1), enabled(2) }

            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
            "The subscriber entry requires (enabled) or does not
            require (disabled) a login password."

            ::= { subsEntry 3 }

password OBJECT-TYPE
```

```
                SYNTAX  DisplayString (SIZE (0..80))
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
                "The login password string of the subscriber entry."

                ::= { subsEntry 4 }

    securityLevel OBJECT-TYPE
                SYNTAX  INTEGER { user(1), super(2) }

                ACCESS  read-write
                STATUS  mandatory
                DESCRIPTION
                "The security level of the subscriber determines what
                privilaged modes they can use."

                ::= { subsEntry 5 }

    maxConnections OBJECT-TYPE
                SYNTAX  INTEGER (1..255)
                ACCESS  read-write
                STATUS  mandatory
                DESCRIPTION
                "The maximum number of the subscriber connections allowed
                for this subscriber entry."

                ::= { subsEntry 6 }

    currentConnections OBJECT-TYPE
                SYNTAX  Gauge
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
                "The number of current connections for the subscriber
                entry."
                ::= { subsEntry 7 }

    activeUserIndex OBJECT-TYPE
                SYNTAX  INTEGER (0..255)
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
                "The active index number of the subscriber entry (0 =
                none)."

                ::= { subsEntry 8 }
```

```
portSecurityList OBJECT-TYPE
              SYNTAX  DisplayString (SIZE (0..80))
              ACCESS  read-write
              STATUS  mandatory
              DESCRIPTION
              "The list of port numbers the subscriber entry is allowed
              access to."

              ::= { subsEntry 9 }

telnetAccess OBJECT-TYPE
              SYNTAX  INTEGER { deny(1), allow(2) }
              ACCESS  read-write
              STATUS  mandatory
              DESCRIPTION
              "The access of the subscriber entry via telnet is allowed
              or denied."

              ::= { subsEntry 10 }


sshAccess OBJECT-TYPE
              SYNTAX  INTEGER { deny(1), allow(2) }
              ACCESS  read-write
              STATUS  mandatory
              DESCRIPTION
              "The access of the subscriber entry via SSH is allowed or
              denied."

              ::= { subsEntry 11 }

guiAccess OBJECT-TYPE
              SYNTAX  INTEGER { deny(1), allow(2) }
              ACCESS  read-write
              STATUS  mandatory
              DESCRIPTION
              "The access of the subscriber entry via the GUI is
              allowed or denied."

              ::= { subsEntry 12 }

consoleAccess OBJECT-TYPE
              SYNTAX  INTEGER { deny(1), allow(2) }
              ACCESS  read-write
              STATUS  mandatory
```

```
                DESCRIPTION
                "The access of the subscriber entry to a Console port is
                allowed or denied."

                ::= { subsEntry 13 }

    dialback OBJECT-TYPE
                SYNTAX  INTEGER { disabled(1), enabled(2) }
                ACCESS  read-write
                STATUS  mandatory
                DESCRIPTION
                "The subscriber entry has the dialback feature enabled or
                disabled."

                ::= { subsEntry 14 }

    dialbackNumber OBJECT-TYPE
                SYNTAX  DisplayString (SIZE (0..15))
                ACCESS  read-write
                STATUS  mandatory
                DESCRIPTION
                "The phone number the subscriber entry will dial back if
                the dialback feature is enabled."

                ::= { subsEntry 15 }

    dialbackRetry OBJECT-TYPE
                SYNTAX  INTEGER (0..255)
                ACCESS  read-write
                STATUS  mandatory
                DESCRIPTION
                "The number of times dialback will be tried for the
                subscriber entry."

                ::= { subsEntry 16 }

    useMenu OBJECT-TYPE
                SYNTAX  INTEGER { disabled(1), enabled(2) }
                ACCESS  read-write
                STATUS  mandatory
                DESCRIPTION
                "The menu feature of the subscriber entry is enabled or
                disabled."

                ::= { subsEntry 17 }
```

```
menuName OBJECT-TYPE
              SYNTAX  DisplayString (SIZE (0..31))
              ACCESS  read-write
              STATUS  mandatory
              DESCRIPTION
              "The menu name for the subscriber entry."

              ::= { subsEntry 18 }

prompt OBJECT-TYPE
              SYNTAX  DisplayString (SIZE (0..15))
              ACCESS  read-write
              STATUS  mandatory
              DESCRIPTION
              "The prompt text of the subscriber entry."

              ::= { subsEntry 19 }

termType OBJECT-TYPE
              SYNTAX  INTEGER {ansi(1), vt100(2)}
              ACCESS  read-write
              STATUS  mandatory
              DESCRIPTION
              "The terminal type of the subscriber entry."

              ::= { subsEntry 20 }

-- sessionMode OBJECT-TYPE
--               SYNTAX  INTEGER {standard(1), vt5xx(2)}
--               ACCESS  read-write
--               STATUS  mandatory
--               DESCRIPTION
--               "The session mode of the subscriber entry."
--
--               ::= { subsEntry 21 }

idleTimeout OBJECT-TYPE
              SYNTAX  INTEGER (0..255)
              ACCESS  read-write
              STATUS  mandatory
              DESCRIPTION
              "The number of minutes the subscriber entry can be idle
              before it is logged out (0 = no timeout)."

              ::= { subsEntry 21 }
```

```
sessionTimeout OBJECT-TYPE
              SYNTAX  INTEGER (0..255)
              ACCESS  read-write
              STATUS  mandatory
              DESCRIPTION
              "The number of minutes the subscriber entry  session can
              be up before it is logged out."

              ::= { subsEntry 22 }

localSwitch OBJECT-TYPE
              SYNTAX  DisplayString (SIZE (0..2))
              ACCESS  read-write
              STATUS  mandatory
              DESCRIPTION
              "The subscriber entry switch to return to the first
              session."

              ::= { subsEntry 23 }

forwardSwitch OBJECT-TYPE
              SYNTAX  DisplayString (SIZE (0..2))
              ACCESS  read-write
              STATUS  mandatory
              DESCRIPTION
              "The subscriber entry switch to move to next session."

              ::= { subsEntry 24 }

backwardSwitch OBJECT-TYPE
              SYNTAX  DisplayString (SIZE (0..2))
              ACCESS  read-write
              STATUS  mandatory
              DESCRIPTION
              "The subscriber entry switch to move to previous
              session."

              ::= { subsEntry 25 }

pause OBJECT-TYPE
              SYNTAX  INTEGER {disable(1),enable(2)}
              ACCESS  read-write
              STATUS  mandatory
              DESCRIPTION
              "The screen pause setting of the subscriber entry."
```

```
                ::= { subsEntry 26 }

    debugging OBJECT-TYPE
                SYNTAX  INTEGER { disabled(1),enabled(2) }
                ACCESS  read-write
                STATUS  mandatory
                DESCRIPTION
                "The subscriber entry debug feature setting."

                ::= { subsEntry 27 }

    logging OBJECT-TYPE
                SYNTAX  INTEGER { disabled(1),enabled(2) }
                ACCESS  read-write
                STATUS  mandatory
                DESCRIPTION
                "The logging feature setting of the subscriber entry."

                ::= { subsEntry 28 }

    preferredService OBJECT-TYPE
                SYNTAX  DisplayString (SIZE (0..15))
                ACCESS  read-write
                STATUS  mandatory
                DESCRIPTION
                "The prefered service of the subscriber entry."

                ::= { subsEntry 29 }

    dedicatedService OBJECT-TYPE
                SYNTAX  DisplayString (SIZE (0..15))
                ACCESS  read-write
                STATUS  mandatory
                DESCRIPTION
                "The dedicated service of the subscriber entry."

                ::= { subsEntry 30 }

    telnetLineMode OBJECT-TYPE
                SYNTAX  INTEGER {line(1),char(2)}
                ACCESS  read-write
                STATUS  mandatory
                DESCRIPTION
                "The telnet line mode setting of the subscriber entry."

                ::= { subsEntry 31 }
```

```
telnetEscapeChar OBJECT-TYPE
               SYNTAX  OCTET STRING
               ACCESS  read-write
               STATUS  mandatory
               DESCRIPTION
               "The telnet escape character of the subscriber entry."

               ::= { subsEntry 32 }

telnetSendcrlf OBJECT-TYPE
               SYNTAX  INTEGER {cr(1),crlf(2)}
               ACCESS  read-write
               STATUS  mandatory
               DESCRIPTION
               "The telnet send newline setting of the subscriber
               entry."

               ::= { subsEntry 33 }

telnetReceivecrlf OBJECT-TYPE
               SYNTAX  INTEGER {cr(1),crlf(2)}
               ACCESS  read-write
               STATUS  mandatory
               DESCRIPTION
               "The telnet receive newline setting of the subscriber
               entry."

               ::= { subsEntry 34 }

sshCipher OBJECT-TYPE
               SYNTAX  INTEGER {any(1),blowfish(2),tripledes(3)}
               ACCESS  read-write
               STATUS  mandatory
               DESCRIPTION
               "The SSH cipher type of the subscriber entry."

               ::= { subsEntry 35 }

remoteSshName OBJECT-TYPE
               SYNTAX  DisplayString (SIZE (0..63))
               ACCESS  read-write
               STATUS  mandatory
               DESCRIPTION
               "The name sent to the remote SSH peer of the subscriber
               entry."
```

```
                ::= { subsEntry 36 }

    sshPort OBJECT-TYPE
                SYNTAX  INTEGER (1..65535)
                ACCESS  read-write
                STATUS  mandatory
                DESCRIPTION
                "The TCP port number used for SSH connections by the
                subscriber entry."

                ::= { subsEntry 37 }

    sshLogLevel OBJECT-TYPE
                SYNTAX  INTEGER
    {quiet(1),info(2),fatal(3),error(4),verbose(5),debug(6)}
                ACCESS  read-write
                STATUS  mandatory
                DESCRIPTION
                "The SSH logging level of the subscriber entry."

                ::= { subsEntry 38 }

    ---- Dynamic Subscriber Table
    --
             sizeOfDynSubscriberTable OBJECT-TYPE
                  SYNTAX  INTEGER (0..65534)
                  ACCESS  not-accessible
                  STATUS  mandatory
                  DESCRIPTION
                  "The size of the dynSubscriberTable."
                  ::= { terminalServer 3 }

             dynSubscriberTable OBJECT-TYPE
                  SYNTAX  SEQUENCE OF DynSubsEntry
                  ACCESS  not-accessible
                  STATUS  mandatory
                  DESCRIPTION
                  "A list of subscriber entries."
                  ::= { terminalServer 4 }

              dynSubsEntry OBJECT-TYPE
                  SYNTAX  DynSubsEntry
                  ACCESS  not-accessible
                  STATUS  mandatory
                  DESCRIPTION
```

```
        "A dynamic subscriber entry containing objects for a
        particular dynamic subscriber."
        INDEX   { dynindex }
        ::= { dynSubscriberTable 1 }

    DynSubsEntry ::=
        SEQUENCE {
            dynindex
                INTEGER,
            subName
                DisplayString,
            devName
                DisplayString,
            devType
                INTEGER,
            dynidleTimeout
                INTEGER,
            dynsessionTimeout
                INTEGER,
            ipAddr
                IpAddress,
            port
                INTEGER,
            protocol
                INTEGER,
            dynprompt
                DisplayString,
            dyntermType
                DisplayString,
            usePpp
                INTEGER,
            dyndialback
                INTEGER,
            remoteLogin
                INTEGER,
            sesActive
                Gauge,
            dynpause
                INTEGER,
            security
                INTEGER,
            totalTransmittedBytes
                Counter,
            totalReceivedBytes
                Counter,
            startTime
                TimeTicks,
```

```
                    kill
                        INTEGER
    }

    dynindex OBJECT-TYPE
                SYNTAX  INTEGER (1..65534)
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
                "A unique index value for each dynamic subscriber."
                ::= { dynSubsEntry 1 }

    subName OBJECT-TYPE
                SYNTAX  DisplayString (SIZE (0..80))
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
                "The name of the dynamic subscriber entry."

                ::= { dynSubsEntry 2 }

    devName OBJECT-TYPE
                SYNTAX  DisplayString (SIZE (0..80))
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
                "The device name of the dynamic subscriber entry."

                ::= { dynSubsEntry 3 }

    devType OBJECT-TYPE
                SYNTAX  INTEGER {physical(1),virtual(2)}
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
                "The device name of the dynamic subscriber entry."

                ::= { dynSubsEntry 4 }

    dynidleTimeout OBJECT-TYPE
                SYNTAX  INTEGER (0..255)
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
                "The number of minutes the dynamic subscriber entry can
                be idle before it is logged out (0 = no timeout)."
```

```
                ::= { dynSubsEntry 5 }


        dynsessionTimeout OBJECT-TYPE
                SYNTAX  INTEGER (0..255)
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
                "The number of minutes the dynamic subscriber entry
                session can be up before it is logged out."

                ::= { dynSubsEntry 6 }

        ipAddr OBJECT-TYPE
                SYNTAX  IpAddress
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
                "The IP address of the dynamic subscriber entry."

                ::= { dynSubsEntry 7 }

        port OBJECT-TYPE
                SYNTAX  INTEGER (0..255)
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
                "The prompt text of the dynamic subscriber entry."

                ::= { dynSubsEntry 8 }

        protocol OBJECT-TYPE
                SYNTAX  INTEGER
        {console(1),serial(2),gui(3),udp(4),telnet(5),ssh(6)}
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
                "The prompt text of the dynamic subscriber entry."

                ::= { dynSubsEntry 9 }

        dynprompt OBJECT-TYPE
                SYNTAX  DisplayString (SIZE (0..15))
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
                "The prompt text of the dynamic subscriber entry."
```

```
                ::= { dynSubsEntry 10 }

    dyntermType OBJECT-TYPE
                SYNTAX  DisplayString (SIZE (0..80))
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
                "The terminal type of the dynamic subscriber entry."

                ::= { dynSubsEntry 11 }

    usePpp OBJECT-TYPE
                SYNTAX  INTEGER { disabled(1), enabled(2) }
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
                "The dynamic subscriber entry uses PPP (enabled) or does
                not use PPP (disabled)."

                ::= { dynSubsEntry 12 }

    dyndialback OBJECT-TYPE
                SYNTAX  INTEGER { disabled(1), enabled(2) }
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
                "The dynamic subscriber entry has the dialback feature
                enabled or disabled."

                ::= { dynSubsEntry 13 }

    remoteLogin OBJECT-TYPE
                SYNTAX  INTEGER { disabled(1), enabled(2) }
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
                "The dynamic subscriber entry is a remote login."

                ::= { dynSubsEntry 14 }

    sesActive OBJECT-TYPE
                SYNTAX  Gauge
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
                "The dynamic subscriber entry has active sessions."
```

```
                ::= { dynSubsEntry 15 }

    dynpause OBJECT-TYPE
                SYNTAX  INTEGER {disable(1),enable(2)}
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
                "The screen pause setting of the dynamic subscriber
                entry."

                ::= { dynSubsEntry 16 }

    security OBJECT-TYPE
                SYNTAX  INTEGER {user(1),super(2)}
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
                "The security setting of the dynamic subscriber entry."

                ::= { dynSubsEntry 17 }

    totalTransmittedBytes OBJECT-TYPE
                SYNTAX  Counter
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
                "The total transmitted bytes of the dynamic subscriber
                entry."

                ::= { dynSubsEntry 18 }

    totalReceivedBytes OBJECT-TYPE
                SYNTAX  Counter
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
                "The total received bytes of the dynamic subscriber
                entry."

                ::= { dynSubsEntry 19 }

    startTime OBJECT-TYPE
                SYNTAX  TimeTicks
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
```

```
                "The start time of the dynamic subscriber entry."

                ::= { dynSubsEntry 20 }

    kill OBJECT-TYPE
                SYNTAX  INTEGER {kill(1)}
                ACCESS  read-write
                STATUS  mandatory
                DESCRIPTION
                "The dynamic subscriber entry is active or killed."

                ::= { dynSubsEntry 21 }

     END        -- end of IN-REACH-Subscriber
```

LX Broadcast Group MIBs

```
    -- $Revision: 1.3 $

               MRV-LX-BROADCASTGROUP-MIB DEFINITIONS ::= BEGIN

    --
    -- Broadcast Group MIB Text File
    --
    -- Date: Wed. Feb.12, 2003 user: DA
    -- File created using EMACS
    -- By: DA
    --
    -- FILE: LX-BR-GROUP-MIB.txt
    --
    --
    --

                        IMPORTS
                                Counter, Gauge, TimeTicks, IpAddress
                                        FROM RFC1155-SMI
                                DisplayString, sysLocation
                                        FROM RFC1213-MIB
                                OBJECT-TYPE
                                        FROM RFC-1212;
              --                TRAP-TYPE
                --                          FROM RFC-1215
                --                charPortIndex
                --                        FROM RFC1316-MIB
                --                rs232InSigState, rs232OutSigState,
    rs232PortIndex
                --                        FROM RFC1317-MIB;
```

```
            -- Define OIDs

            -- ccitt            OBJECT IDENTIFIER ::= { 0 }
            -- null             OBJECT IDENTIFIER ::= { ccitt 0 }

             iso              OBJECT IDENTIFIER ::= { 1 }
             org              OBJECT IDENTIFIER ::= { iso 3 }
             dod              OBJECT IDENTIFIER ::= { org 6 }
             internet         OBJECT IDENTIFIER ::= { dod 1 }
             directory        OBJECT IDENTIFIER ::= { internet 1 }
             mgmt             OBJECT IDENTIFIER ::= { internet 2 }
             experimental     OBJECT IDENTIFIER ::= { internet 3 }
             private          OBJECT IDENTIFIER ::= { internet 4 }
             enterprises      OBJECT IDENTIFIER ::= { private 1 }

            mrvInReachProductDivision OBJECT IDENTIFIER ::=
{enterprises 33}
            lxagent            OBJECT IDENTIFIER ::=
{mrvInReachProductDivision 50}

            terminalServer   OBJECT IDENTIFIER ::= { lxagent 1 }

            -- MRV In-Reach LX Broadcast Groups

            -- This is a MIB module for all MRV In-Reach LX systems that
            -- implement broadcast groups.
            --
            -- NOTE: The broadcast groups are actually under the interface
            -- level, but are currently locked on interface 1. This may
            -- change at sometime in the future and require corresponding
            -- changes to this MIB as well.
            --
            -- Copyright 2003 MRV Communications, Inc. All Rights Reserved.
            -- Reproduction of this document is authorized on
            -- condition that this copyright notice is included.
            -- This MIB document embodies MRV Communications, Inc.'s
            -- proprietary intellectual property.  MRV Communications, Inc.
            -- retains all title and ownership in this MIB, including any
            -- revisions.
            --
            -- It is MRV Communications, Inc.'s intent to encourage the
            -- widespread use of this MIB in connection with the management
            -- of MRV Communications, Inc. products. MRV Communications,
            -- Inc. grants vendors, end-users, and other interested parties
            -- a non-exclusive license to use this MIB in connection with
            -- the management of MRV Communications, Inc. products.
```

```
 --
 -- This MIB document is supplied "AS IS," and MRV
 -- Communications, Inc. makes no warranty, either express
 -- or implied, as to the use operation, condition, or
 -- performance of the MIB.

 -- Implementation of this group is mandatory for all MRV
 -- In-Reach LX systems that implement broadcast groups.

 -- the lxBroadcastGroup table

sizeOfBroadcastGroupTable OBJECT-TYPE
     SYNTAX  INTEGER (1..65534)
     ACCESS  not-accessible
     STATUS  mandatory
     DESCRIPTION
     "The size of the subscriberTable."
     ::= { terminalServer 6 }

 lxBroadcastGroupTable OBJECT-TYPE
     SYNTAX  SEQUENCE OF LxBroadcastGroupEntry
     ACCESS  not-accessible
     STATUS  mandatory
     DESCRIPTION
     "A list of broadcast group parameters."
     ::= { terminalServer 7 }

 lxBroadcastGroupEntry OBJECT-TYPE
     SYNTAX  LxBroadcastGroupEntry
     ACCESS  not-accessible
     STATUS  mandatory
     DESCRIPTION
     "Broadcast group information."
     INDEX { lxBroadcastGroupIndex }
     ::= { lxBroadcastGroupTable 1 }

 LxBroadcastGroupEntry ::=
     SEQUENCE {
         lxBroadcastGroupIndex
             INTEGER,
         lxBroadcastGroupMode
             INTEGER,
         lxBroadcastGroupState
       INTEGER,
         lxBroadcastGroupAsyncMasterTimestampList
       DisplayString,
     lxBroadcastGroupAsyncMasterNoTimestampList
```

```
  DisplayString,
lxBroadcastGroupTcpMasterTimestampList
  DisplayString,
lxBroadcastGroupTcpMasterNoTimestampList
  DisplayString,
lxBroadcastGroupAsyncSlaveDiscardList
  DisplayString,
lxBroadcastGroupAsyncSlaveNoDiscardList
  DisplayString,
lxBroadcastGroupAsyncSlaveLocalEchoList
  DisplayString,
lxBroadcastGroupAsyncSlaveNoLocalEchoList
  DisplayString,
lxBroadcastGroupTcpSlaveDiscardList
  DisplayString,
lxBroadcastGroupTcpSlaveNoDiscardList
  DisplayString,
lxBroadcastGroupTcpSlaveLocalEchoList
  DisplayString,
lxBroadcastGroupTcpSlaveNoLocalEchoList
  DisplayString,
lxBroadcastGroupAsyncMasterTimestampPort
  INTEGER,
lxBroadcastGroupAsyncMasterNoTimestampPort
  INTEGER,
lxBroadcastGroupTcpMasterTimestampPort
  INTEGER,
lxBroadcastGroupTcpMasterNoTimestampPort
  INTEGER,
lxBroadcastGroupAsyncSlaveDiscardPort
  INTEGER,
lxBroadcastGroupAsyncSlaveNoDiscardPort
  INTEGER,
lxBroadcastGroupAsyncSlaveLocalEchoPort
  INTEGER,
lxBroadcastGroupAsyncSlaveNoLocalEchoPort
    INTEGER,
lxBroadcastGroupTcpSlaveDiscardPort
  INTEGER,
lxBroadcastGroupTcpSlaveNoDiscardPort
  INTEGER,
lxBroadcastGroupTcpSlaveLocalEchoPort
  INTEGER,
lxBroadcastGroupTcpSlaveNoLocalEchoPort
  INTEGER
}
```

```
            lxBroadcastGroupIndex OBJECT-TYPE
              SYNTAX INTEGER
              ACCESS read-only
              STATUS mandatory
              DESCRIPTION
             "An index value that uniquely identifies a broadcast
             group."
              ::= { lxBroadcastGroupEntry 1 }

              lxBroadcastGroupMode OBJECT-TYPE
                  SYNTAX INTEGER { line(1), character(2) }
                  ACCESS read-write
                  STATUS mandatory
                  DESCRIPTION
                  "A mode value for the broadcast group. The value
                  'line' means that the group is active and broadcasting
                  data in line mode whereas the value 'character' means
                  that the group is active and broadcasting data in
                  character mode."
                  DEFVAL { line }
                  ::= { lxBroadcastGroupEntry 2 }

              lxBroadcastGroupState OBJECT-TYPE
                  SYNTAX INTEGER { disabled(1), enabled(2) }
                  ACCESS read-write
                  STATUS mandatory
                  DESCRIPTION
                  "A state value for the broadcast group. The value
                  'enabled' means that the group is active and
                  broadcasting data whereas 'disabled' means that the
                  group is not active."
                  DEFVAL { disabled }
                  ::= { lxBroadcastGroupEntry 3 }
--
-- object types to display the master & slave port lists in a
-- specific broadcast group.
--
              lxBroadcastGroupAsyncMasterTimestampList OBJECT-TYPE
                SYNTAX DisplayString
                ACCESS read-only
                STATUS mandatory
                DESCRIPTION
                "A list of the Async Master Broadcast Ports with
                timestamp enabled for this group."
                  ::= { lxBroadcastGroupEntry 4 }

              lxBroadcastGroupAsyncMasterNoTimestampList OBJECT-TYPE
```

```
  SYNTAX DisplayString
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
  "A list of the Async Master Broadcast Ports without
  timestamp enabled for this group."
    ::= { lxBroadcastGroupEntry 5 }

lxBroadcastGroupTcpMasterTimestampList OBJECT-TYPE
  SYNTAX DisplayString
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
  "A list of the Tcp Master Broadcast Ports with timestamp
   enabled for this group."
    ::= { lxBroadcastGroupEntry 6 }

lxBroadcastGroupTcpMasterNoTimestampList OBJECT-TYPE
  SYNTAX DisplayString
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
  "A list of the Tcp Master Broadcast Ports without
  timestamp enabled for this group."
    ::= { lxBroadcastGroupEntry 7 }

lxBroadcastGroupAsyncSlaveDiscardList OBJECT-TYPE
  SYNTAX DisplayString
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
  "A list of the Async Slave Broadcast Ports with discard
  enabled for this group."
    ::= { lxBroadcastGroupEntry 8 }

lxBroadcastGroupAsyncSlaveNoDiscardList OBJECT-TYPE
  SYNTAX DisplayString
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
  "A list of the Async Slave Broadcast Ports without
  discard enabled for this group."
    ::= { lxBroadcastGroupEntry 9 }

lxBroadcastGroupAsyncSlaveLocalEchoList OBJECT-TYPE
  SYNTAX DisplayString
  ACCESS read-only
```

```
  STATUS mandatory
  DESCRIPTION
  "A list of the Async Slave Broadcast Ports with local
  echo enabled for this group."
    ::= { lxBroadcastGroupEntry 10 }

lxBroadcastGroupAsyncSlaveNoLocalEchoList OBJECT-TYPE
  SYNTAX DisplayString
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
  "A list of the Async Slave Broadcast Ports without local
  echo enabled for this group."
    ::= { lxBroadcastGroupEntry 11 }

lxBroadcastGroupTcpSlaveDiscardList OBJECT-TYPE
  SYNTAX DisplayString
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
  "A list of the Tcp Slave Broadcast Ports with discard
  enabled for this group."
    ::= { lxBroadcastGroupEntry 12 }

lxBroadcastGroupTcpSlaveNoDiscardList OBJECT-TYPE
  SYNTAX DisplayString
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
  "A list of the Tcp Slave Broadcast Ports without discard
  enabled for this group."
    ::= { lxBroadcastGroupEntry 13 }

lxBroadcastGroupTcpSlaveLocalEchoList OBJECT-TYPE
  SYNTAX DisplayString
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
  "A list of the Tcp Slave Broadcast Ports with local echo
  enabled for this group."
    ::= { lxBroadcastGroupEntry 14 }

lxBroadcastGroupTcpSlaveNoLocalEchoList OBJECT-TYPE
  SYNTAX DisplayString
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
```

```
                "A list of the Tcp Slave Broadcast Ports without local
                echo enabled for this group."
                  ::= { lxBroadcastGroupEntry 15 }
--
-- object types to configure the master & slave ports in a
-- specific broadcast group.
--
            lxBroadcastGroupAsyncMasterTimestampPort OBJECT-TYPE
              SYNTAX INTEGER
              ACCESS write-only
              STATUS mandatory
              DESCRIPTION
              "An  Async Master Broadcast Port with timestamp
              enabled for this group."
                ::= { lxBroadcastGroupEntry 16 }

            lxBroadcastGroupAsyncMasterNoTimestampPort OBJECT-TYPE
              SYNTAX INTEGER
              ACCESS write-only
              STATUS mandatory
              DESCRIPTION
              "An Async Master Broadcast Port without timestamp
              enabled for this group."
                ::= { lxBroadcastGroupEntry 17 }

            lxBroadcastGroupTcpMasterTimestampPort OBJECT-TYPE
              SYNTAX INTEGER
              ACCESS write-only
              STATUS mandatory
              DESCRIPTION
              "A Tcp Master Broadcast Port with timestamp
              enabled for this group."
                ::= { lxBroadcastGroupEntry 18 }

            lxBroadcastGroupTcpMasterNoTimestampPort OBJECT-TYPE
              SYNTAX INTEGER
              ACCESS write-only
              STATUS mandatory
              DESCRIPTION
              "A Tcp Master Broadcast Port without timestamp
              enabled for this group."
                ::= { lxBroadcastGroupEntry 19 }

            lxBroadcastGroupAsyncSlaveDiscardPort OBJECT-TYPE
              SYNTAX INTEGER
              ACCESS write-only
              STATUS mandatory
```

```
  DESCRIPTION
  "An Async Slave Broadcast Port with discard
  enabled for this group."
    ::= { lxBroadcastGroupEntry 20 }

lxBroadcastGroupAsyncSlaveNoDiscardPort OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS write-only
  STATUS mandatory
  DESCRIPTION
  "An Async Slave Broadcast Port without discard
  enabled for this group."
    ::= { lxBroadcastGroupEntry 21 }

lxBroadcastGroupAsyncSlaveLocalEchoPort OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS write-only
  STATUS mandatory
  DESCRIPTION
  "An Async Slave Broadcast Port with local echo
  enabled for this group."
    ::= { lxBroadcastGroupEntry 22 }

lxBroadcastGroupAsyncSlaveNoLocalEchoPort OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS write-only
  STATUS mandatory
  DESCRIPTION
  "An Async Slave Broadcast Port without local echo
  enabled for this group."
    ::= { lxBroadcastGroupEntry 23 }

lxBroadcastGroupTcpSlaveDiscardPort OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS write-only
  STATUS mandatory
  DESCRIPTION
  "A Tcp Slave Broadcast Port with discard
  enabled for this group."
    ::= { lxBroadcastGroupEntry 24 }

lxBroadcastGroupTcpSlaveNoDiscardPort OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS write-only
  STATUS mandatory
  DESCRIPTION
  "A Tcp Slave Broadcast Port without discard
```

```
                    enabled for this group."
                    ::= { lxBroadcastGroupEntry 25 }

                lxBroadcastGroupTcpSlaveLocalEchoPort OBJECT-TYPE
                  SYNTAX INTEGER
                  ACCESS write-only
                  STATUS mandatory
                  DESCRIPTION
                  "A Tcp Slave Broadcast Port with local echo
                  enabled for this group."
                    ::= { lxBroadcastGroupEntry 26 }

                lxBroadcastGroupTcpSlaveNoLocalEchoPort OBJECT-TYPE
                  SYNTAX INTEGER
                  ACCESS write-only
                  STATUS mandatory
                  DESCRIPTION
                  "A Tcp Slave Broadcast Port without local echo
                  enabled for this group."
                    ::= { lxBroadcastGroupEntry 27 }
--
--
--
            END  -- End of MRV-LX-BROADCASTGROUP-MIB
```

LX Series-Supported RFCs

The LX also implements the MIBs defined by the following RFCs:

- 1213 - MIB2
- 1659 - RS232
- 1907 - SNMPv2
- 1696 - Modem MIB
- 1658 - Character Device
- 1472 - PPP Security Protocols
- 1471 - PPP Link Control Protocol
- 1473 - PPP IP Network Control Protocol
- 2574 - User Based Security Model for SNMPv3
- 2575 - View-based Access Control Model for SNMPv3

Standard MIBs

These files are standard RFC documents, as defined by the Internet Engineering Task Force (IETF). They are provided as *information only*, because they are used by MRV Communications in building Concatenated MIBs. The RFCs listed here are used *solely* by MRV Communications.

Note: This is ***not*** to be considered an official repository of Requests for Comments (RFCs). It is considered a living document, whereby it is subject to change at any time.

RFC1155.SMI "Structure of Management Information"
RFC1212.SMI "Concise MIB Format"
RFC1213.MIB "MIB II"
RFC1215.SMI "Concise Trap Format"
RFC1215.TRP "Trap Definitions"
RFC1229.MIB "Extensions to Generic Interface MIB"
RFC1284.MIB "MIB for Ethernet-like objects"
RFC1317.MIB "MIB for RS-232-like Hardware Devices"
RFC1354.MIB "IP Forwarding Table MIB"
RFC1398.MIB "Definitions of Managed Objects for Ethernet-like Interface Types"
RFC1471.MIB "Link Control Protocol of PPP"
RFC1472.MIB "Security Protocols of PPP"
RFC1473.MIB "IP Network Control Protocol of PPP"
RFC1573A.MIB "V2 evolution of MIB-II, part A"
RFC1573B.MIB "V2 evolution of MIB-II, part B"