



THE SNMP SHELL

September 2000





THE SNMP SHELL

A	REFERENCE	5
1	The SNMP shell	6
1.1	SNMP Explained	6
1.1.1	Overview	6
1.1.2	The MIB	7
1.2	SNMP Shell Overview	12
1.2.1	The Shell Prompt	12
1.2.2	Command Line Editing	13
1.2.3	Object Types	14
1.2.4	Shell Commands	15
1.2.5	External Commands	27
1.3	BinTec router System Tables	41
1.3.1	Short vs. Long Names	43
1.3.2	Creating Table Entries	44
1.3.3	Deleting Table Entries	45
1.3.4	Editing Table Entries	46
1.4	BinTec router Interfaces	48
1.4.1	Special Interfaces	49
1.4.2	Hardware Interfaces	50
1.4.3	Software Interfaces	53
1.5	BinTec router Configuration Files	55
1.5.1	Managing FLASH files	55
1.5.2	Transferring Files with TFTP	61
1.5.3	Transferring Files with XMODEM via Serial Port	66

1.5.4 Rebooting the System

68

REFERENCE

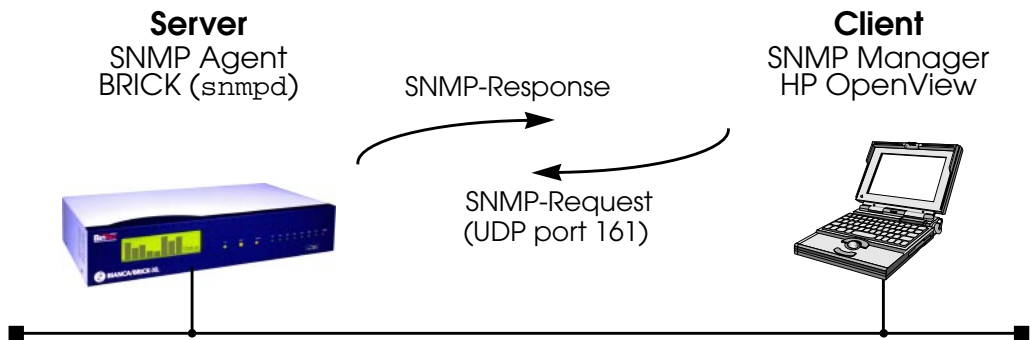
1 The SNMP shell

1.1 SNMP Explained

1.1.1 Overview

[SNMP \(Simple Network Management Protocol\)](#), the successor to SGMP (Simple Gateway Monitoring Protocol), is used to manage network devices (workstation, terminal server, printer, bridge, hub). SNMP is an Application Layer protocol and uses the underlying [UDP \(User Datagram Protocol\)](#) as its transport medium; SNMP defines the rules of communication between an SNMP Manager and SNMP Agent allowing a network administrator to “watch” and/or control individual devices by viewing/changing operational settings stored on the managed device.

SNMP-based network management is a Client-Server system; however, the terms *Manager* and *Agent* are misleading in this context.



SNMP can be seen as a simple asynchronous **request-response** protocol. Messages are passed via UDP (normally port 161) and are binary in format. The SNMP Manager re-

quests information from a specific device and an SNMP Agent (running on the device) authenticates the requester and responds with the requested information.

As mentioned above, different types of network devices may be managed via SNMP. Inherently, such systems have very different types of operational settings (comparing say a router to a printer). SNMP is not concerned with the contents of the messages being sent but with the methods used to obtain and change the settings on the remote devices. This is why SNMP is referred to as a simple protocol; because all network management functions basically boil down to a few basic operations. Operations available to the manager and agent processes are as follows.

Operations available to the SNMP Manager:

- get-request* Requests the value associated with a specific variable.
- get-next-request* Requests the next value associated with a variable that comprises a list of elements.
- set-request* Sets or changes the value of a specific variable.

Operations available to the SNMP Agent:

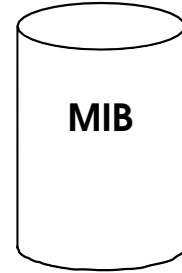
- get-response* Returns the value of a variable associated with a previous get-request or get-next-request message.
- trap* Reports the occurrence of a fault condition, or other important relevant information.

1.1.2 The MIB

The [MIB \(Management Information Base\)](#) defines objects, often called MIB objects, that can be managed (i.e., queried, changed, created) for a particular device via SNMP. Objects

per se are simply templates that define characteristics about a particular device and would include such things as:

- **Object Names**—How can the object be identified?
- **Object Descriptions**—How does the object's assigned value relate to the overall operational state of the device?
- **Object Access**—Who is allowed to change the object's setting?
- **Object Types**—Can the object's value be changed?
- **Object Ranges**—What values can be assigned to the object?



Consider IP routing for a multiprotocol router such as the BinTec router. An IP route consists of several variables including at a minimum: Destination IP Address, IP Netmask, IP Metric, and Router Interface. Each of these items would be defined separately in the MIB. An example is an IP route's Next Hop object; though commonly referred to as *ipNextHop* its complete name is: **.iso.org.dod.internet.management.mib2.ip.iproutetable.ipNextHop**.

And in numerical form: .1.3.6.1.2.1.4.21.1.7 (see [MIB Structure](#)).

Also, a router's routing tables consist of multiple entries; thus multiple instances of the same object type would exist on a running system. This is a fundamental concept and means that the router needs a mechanism to uniquely identify a specific instance of an object. The naming structure used by the MIB provides this mechanism by associating a MIB object with a local number and is called **Instance Identification**. When managing an IP router via SNMP it is *instances* of objects that are being manipulated.

SNMP Managers

Intelligent SNMP managers can communicate effectively with devices when the structure of MIB objects are known to it. ASCII files containing descriptions of MIB objects supported by a device are normally provided by the device's manufacturer and can be imported (or compiled) into SNMP manager applications.

MIB Structure

MIB objects have a hierarchical naming structure that forces every object to be unique to all other objects. This hierarchy is similar to a tree structure and is managed by the IANA (Internet Assigned Numbers Authority). Each node in the tree relates to a document that defines objects below that point. In SNMP individual object names are called Object Identifiers, or OIDs.

The figure on the following page shows the tree hierarchy relating to MIB objects implemented on the BIANCA/BRICK.

Note that OIDs can be referenced in two ways.

- Numerically

Using the numbers assigned by the documents in the tree shown [here](#) a router's IP routing table is defined as object #21 of the ip module in the document that describes mib2.

.1.3.6.1.2.1.4.21

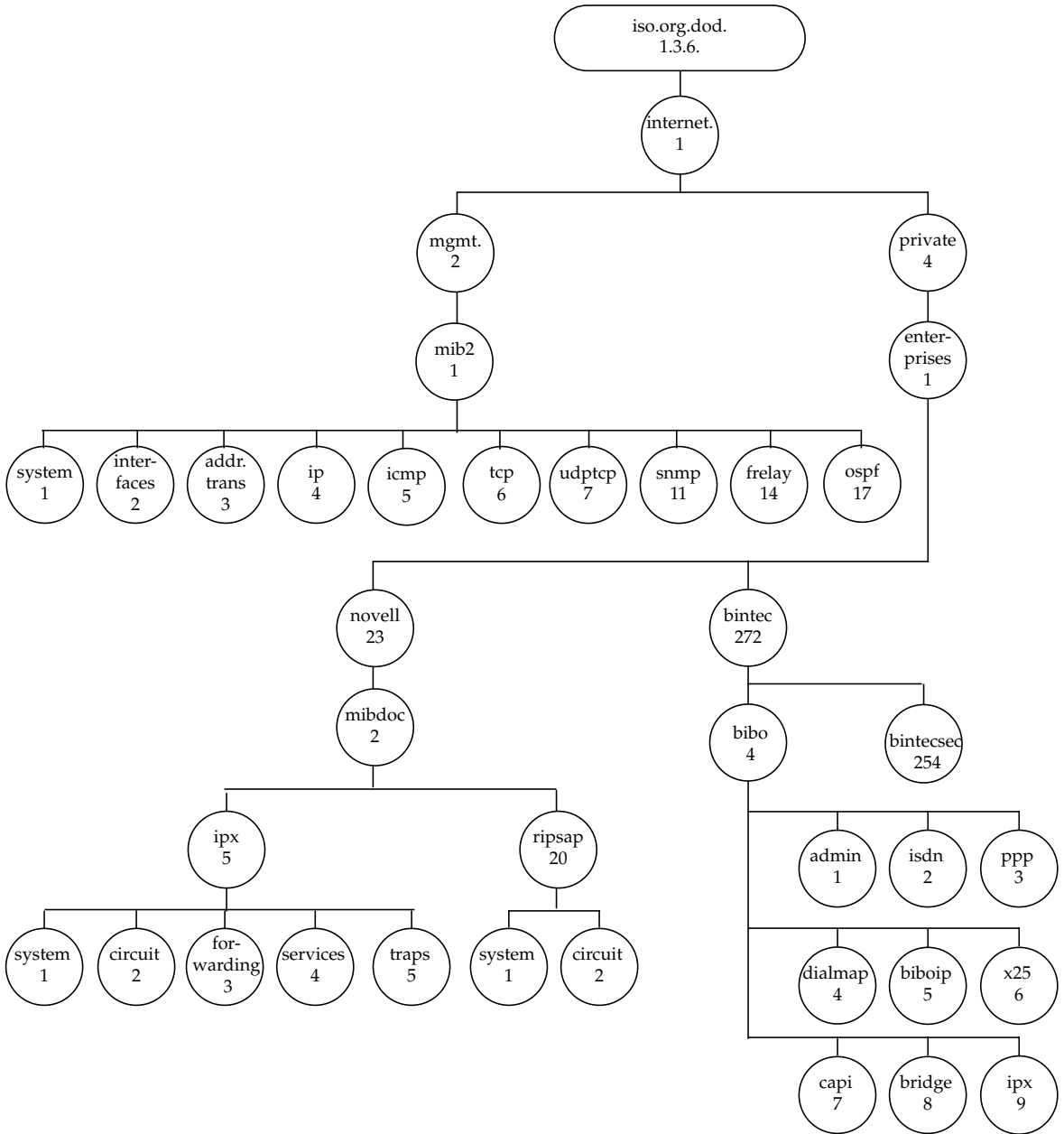
- Textually

Text names can also be used to identify objects. The router's IP route table would have the symbolic OID of:

.iso.org.dod.internet.management.ip.iproutetable

Objects under the .iso.org.dod.internet.management tree are standard MIB objects defined by the ISO; enterprises (companies such as router manufacturers and protocol developers) may be assigned subtrees by IANA where their product specific objects can be defined. Since like devices provide similar services, and to provide interoperability between existing SNMP managers most devices support standard MIB objects defined by the ISO (International Organization for Standardization).

For internet routers MIB-2 (defined in RFC 1158) is the current standard and defines such objects as IP routing tables and various IP protocol settings. Multiprotocol routers that support IPX will also support Novell's enterprise MIB definitions.



1.2 SNMP Shell Overview

Along with other routing processes, the BinTec router starts an SNMP Agent (see [SNMP Explained](#)) process when at boot time. You'll see these processes listed to the screen when the system is started (if a console is attached via the serial port). After all processes have been started, a login prompt is presented to the screen.

This login session is what we call the SNMP-shell.

The SNMP shell serves the same purpose as an SNMP Manager application. All of the BinTec router's MIB objects can be managed from this shell. And because the shell is character based, the BinTec router can be accessed remotely over any character-oriented connections such as:

- telnet sessions (from PCs or Workstations)
- HyperTerminal sessions (Windows 95 /Windows NT)
- isdnlogin sessions (isdnlogin command is provided)
- X.25 pad calls (minipad utility is provided)

1.2.1 The Shell Prompt

As shown below the shell prompt consists of two parts separated by a colon.

mybrick: ipRouteTable>

Current **SysName** Current system table.

If the contents of the *sysName* object (system table) is not set, the first part of the prompt defaults to "brick".





Also, as you navigate among MIB objects the prompt will change to reflect the last system table displayed; simi-

lar to the "current working directory" variable used with many UNIX shells.

Knowing the current system table can be very useful when editing MIB objects because it allows you to use an object's short name instead of the complete object name; this is covered in the section [Short vs. Long Names](#).

1.2.2 Command Line Editing

A command line editor is available from the SNMP shell. The command line editor allows you to edit commands on the command line before pressing the <Return> key letting you adjust parameter settings or typing mistakes of previously entered commands. The up, down, right, and left arrow keys can be used as follows.

Key	Meaning
	Command History Moves you backwards through the list of commands entered during this shell session.
	Command History Moves you forwards through the list of commands entered during this shell session.
	Command Editing Moves the cursor backwards through the currently displayed command.
	Command Editing Moves the cursor forwards through the currently displayed command.

The command line editor is always in insert mode. Once the cursor is moved along the command line new characters typed in are inserted at the cursor's location. The <backspace> key can be used to erase characters.

1.2.3 Object Types

Each MIB object has a type associated with it that defines the types of values that it can be assigned. An object's type can be any of the following.

- Integer Value
- Character String
- IP Address
- Object Identifier
- Octet String
- Enumerated Value

For some objects the type of value that it may be assigned will be clear from the object's name. The *Address* variable in the *biboPPPIpAssignTable* is a good example, it accepts an IP address in dot format. You can determine an object's type from the SNMP shell by entering the object's name followed by a ? (no space in between) and pressing <Return>. For example;

```
mybrick::system> ipRouteDest?  
ipRouteDest: (readwrite) IP-address in dot-format (eg. 1.2.3.4)  
  
mybrick::ipRouteTable> ipRouteInfo?  
ipRouteInfo: (readwrite) object identifier in dot format (eg. .1.3.6.1)  
  
mybrick::ipRouteTable> ipRouteType?  
ipRouteType: (readwrite) other (1), delete (2), invalid (2), direct (3), indirect  
  
mybrick::ipRouteTable> biboDialStkMask?  
biboDialStkMask: (readwrite) binary integer (e.g. 0b1101)
```

Objects that accept "integer values" can be set using one of four numbering systems as described below. Note however that some objects only accept numerical values in a specific numbering system such as "binary integers" as shown in the last example above.

Integer Values

When setting MIB objects that accept integer values the four numbering systems shown below may be used.

Numbering System	Prefix	Example Command	Resulting Decimal Value
Decimal	<none>	<i>ipDefaultTTL=10</i>	10
Octal	0	<i>ipDefaultTTL=012</i>	
Hexadecimal	0x	<i>ipDefaultTTL=0xa</i>	
Binary	0b	<i>ipDefaultTTL=0b1010</i>	

In most cases the decimal system is used; when using other numbering systems the above prefixes must be used to identify the appropriate numbering system.

Enumerated Types

Many MIB objects only accept values from a predefined list. These objects are said to be enumerated types. For example the *Compression* object in the *biboPPPTable* can be set to **none**, **v42bis**, or **stac**. No other values are acceptable.

The values for these objects are numbered starting at one with the first value being the objects default value. These numbers can also be used to set an object to the respective enumerated value. This means that the commands **Compression=v42bis** has the same effect as **Compression=2**

1.2.4 Shell Commands

The following commands are available from the SNMP shell.

Command	Usage	Meaning
Help	?	Lists all shell and external commands.

Command	Usage	Meaning
Community	c [<i><community></i>]	Sets/displays current SNMP community.
Group	g [<i><groupnumber></i> <i><groupname></i> *]	Lists all groups or all tables within a group.
List	l	Lists all tables.
Priority	p [<i><high low></i>]	Sets/displays current shell priority setting.
Columns	u [<i><columns></i>]	Sets/displays the number of columns used when displaying table output to screen.
Raw-Mode	x	Toggles shell's raw mode on and off.
Table-Mode	y	Toggles shell's table mode on and off.
Lines	z [<i><lines></i>]	Sets/displays the number of lines used when displaying table output to screen.
Exit	exit	Exits the current SNMP shell.

The help command (?)

Usage: ?

The help command simply lists a summary of all available internal and external shell commands to the screen.

The Community Command (c)

Usage: c [<communityname>]

The community command sets or displays the current SNMP community name to use for SNMP command requests issued from the current shell.

Community names correspond to the password strings configured for the *biboAdmAdminCommunity*, *biboAdmReadCommunity*, and *biboAdmWriteCommunity* objects of *bintecsec*.

For example, if:

1. The current value of *biboAdmAdminCommunity* = bianca AND
2. You are currently logged in as the admin user, AND
3. The community hasn't changed since logging in.

then your current community name (as displayed using c) is bianca.

Changing the community name during an SNMP shell session effectively changes read/write permission for MIB objects. This is similar to the UNIX su command except that no subshell is started.

If you log in as the admin user and change the value of *biboAdmAdminCommunity*, the current community is automatically adjusted to the new value (with one exception as noted below).

NOTE

If you manually changed the community name any time during a shell session, the BinTec router will no longer be able to automatically update the community name upon changes to ***biboAdmAdminCommunity***.

You will have to change the community name manually or log out and log in again using the new value.

The Group Command (g)

Usage: **g** [<groupnumber> | <groupname> | *]

The group command lists BinTec router subsystem groups, or all tables within a specific group to the screen. Options are used as follows:

<groupnumber> Specifies a group whose tables should be listed.

NOTE

The shell interprets integer values according to the format they are entered in. See [Integer Values](#).

If you enter the command **g 08**, the shell interprets the leading 0 as identifying an octal and will report that the group doesn't exist. The command **g 010** would display tables in group eight; as would **g 0b1000**, **g 0xb**, and **g 8**.

<groupname> Specifies a group whose tables should be listed.

Possible group names are those listed using the **g** command without parameters.

* Lists all tables without showing which group they belong to.

The List Command (l)

Usage: **l**

The list command lists all system tables to the screen. Table names are displayed in numerical order grouped by BinTec router subsystem. A table's contents may be listed by entering the table name, or its number; i.e., entering **system** displays the same results as entering **1**.

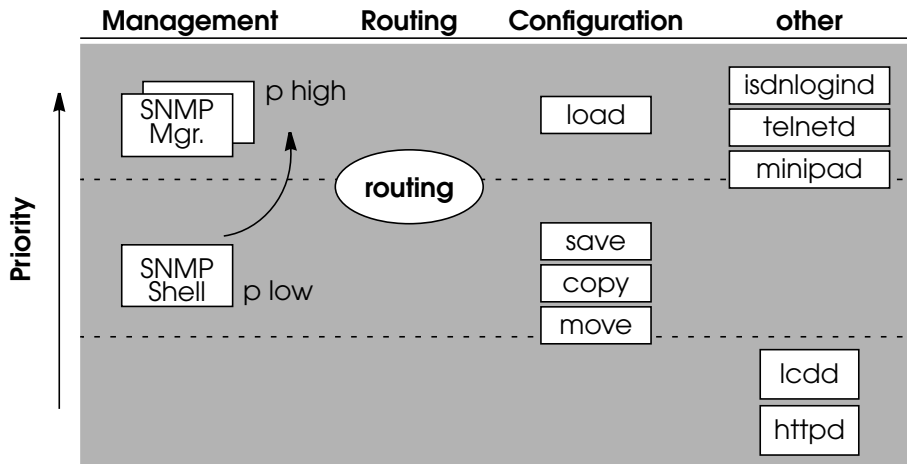
The Priority Command (p)

Usage: **p** [**high** | **low**]

The **p** (priority) command sets the priority (high or low) of the BinTec router's SNMP shell with respect to other system processes.

The specified priority becomes effective for the current shell and all sub-processes started from this shell. When the **p** command is used without arguments the current shell priority is displayed.

By default, the SNMP shell has a lower priority than routing processes which means that an interactive configuration session (setup) will not affect performance on systems with many WAN partners.



Shell-priority and `cmd=save` commands:

If the shell's priority is set to **high** and a configuration is saved, the SNMP shell immediately returns you to the command prompt. This is in contrast to **low** status where the prompt is returned only after the save command is completed. The state of a `cmd=save` command can be verified by displaying the *biboAdmConfigTable*.

Note:

This does not apply to SetupTool sessions. SetupTool always waits for configuration management commands to complete before proceeding.

The Columns Command (`u`)

Usage: `u` [*<columns>*]

The columns command displays or sets the number of columns to use when writing to the screen; by default 79 columns. This is useful in Table Mode when using a non-standard sized terminal window (i.e., bigger than 80x24).

When using Setup Tool, the number of columns doesn't matter, since Setup Tool always displays output for a 79 column terminal window.

The Lines Command (`z`)

Usage: `z` [*<lines>*]

The lines command displays or sets the number of lines to use when writing to the screen; by default 24 lines. This is useful in Table-Mode when using a non standard sized terminal window (i.e., bigger than 80x24).

When in Table-Mode, the shell displays as many "complete" table entries as possible (*<lines required>* ≤ `z` *<lines>*) before prompting the user to continue with "**Press <RETURN> to continue or <q> to quit.**"

The Exit Command (`exit`)

Usage: `exit`

The exit command ends the current SNMP shell session and returns you to a new login prompt. Another way to exit the current shell is to enter Control-d, (holding down the Ctrl key and entering d).

The Raw-Mode Command (x)

Usage: **x**

The Raw-Mode command toggles Raw-Mode on and off. After entering the command, the shell reports which mode it is entering. By default Raw-Mode is off from the SNMP shell.

Raw-Mode means that MIB objects are dumped to the screen in the following format:

```
<MIB OID> . <DB Key> . <Dynamic Number>
( <Type> ) <Value>
```

<i>MIB OID</i>	The MIB Object's OID in numerical format.
<i>DB Key</i>	The DB Key identifies a specific instance of MIB OID and consists of the numerical values of all index variables for the system table.
<i>Dynamic Number</i>	A local number that the BinTec router assigns locally.
<i>Type</i>	The MIB Object's type; integer, string, IP address, etc.
<i>Value</i>	The current value of this instance of MIB OID.

When Raw-Mode is on the current columns, lines, and Table-Mode settings are disregarded. Although the command prompt is not present when Raw-Mode is on, the command-line editor (cursor and backspace keys) can still be used. MIB objects can be set or queried using their numerical or symbolic names. Although this mode is not intended for interactive use it is possible to selectively display and set system variables. The difference between Raw-Mode on and off is shown below.

Raw-Mode on:

```

mybrick:system> x
rawmode on
ipNetToMediaTable
.1.3.6.1.2.1.4.22.1.1.1000.192.168.6.7.0 (Integer) 1000
.1.3.6.1.2.1.4.22.1.2.1000.192.168.6.7.0 (PhysAddress) 0:a0:f9:0:3:f2
.1.3.6.1.2.1.4.22.1.3.1000.192.168.6.7.0 (IpAddress) 192.168.6.7
.1.3.6.1.2.1.4.22.1.4.1000.192.168.6.7.0 (Integer) 4
.1.3.6.1.2.1.4.22.1.1.1000.192.54.54.2.1 (Integer) 1000
.1.3.6.1.2.1.4.22.1.2.1000.192.54.54.2.1 (PhysAddress) 0:0:f1:ab:f2:f3
.1.3.6.1.2.1.4.22.1.3.1000.192.54.54.2.1 (IpAddress) 192.54.54.2
.1.3.6.1.2.1.4.22.1.4.1000.192.54.54.2.1 (Integer) 4

```

Raw-Mode Off (default):

```

x
rawmode off
mybrick:system> ipNetToMediaTable

inx IfIndex(*rw) PhysAddress(rw) NetAddress(*rw) Type(-rw)

00 1000          0:a0:f9:0:3:f2  192.168.6.7     static
01 1000          0:0:f1:ab:f2:f3 192.54.54.2     static

mybrick:ipNetToMediaTable>

```


The Table-Mode Command (y)

Usage: **y**

The Table-Mode command toggles Table-Mode on and off. After entering the command the shell reports which mode it is now entering. By default Table-Mode is on.

When Table-Mode is on the shell formats output using the current lines, and columns settings. Thus when a table's contents are displayed as many "complete" table entries are displayed as possible. The table's column names are displayed followed by the rows with each row separated by a blank line. If the table consists of more entries than can be displayed to the window (see the [lines](#) command), the user is prompted to continue.

Table-Mode On (default):

```
zeusbox:system> ipRouteTable

inx Dest(*rw)  Iifindex(rw)  Metric1(rw)  Metric2(rw)
Metric3(rw)  Metric4(rw)  NextHop(rw)  Type(-rw)
Proto(ro)    Age(rw)       Mask(rw)     Metric5(rw)
Info(ro)

00 192.168.12.0  1000          0             -1
-1              -1            192.168.12.20 direct
netmgmt        28674         255.255.255.128 -1
.0.0

01 192.168.6.0   1000          0             -1
-1              -1            0.0.0.0      indirect
netmgmt        28674         255.255.255.0 -1
.0.0

02 16.0.0.30     10001         0             -1
-1              -1            16.0.0.30    direct
netmgmt        28675         255.255.255.0 -1
.0.0

Press <RETURN> to continue or <q> to quit.
```

Table-Mode Off:

```

15: ipRouteMetric2.192.168.4.128.15( rw): -1
15: ipRouteMetric3.192.168.4.128.15( rw): -1
15: ipRouteMetric4.192.168.4.128.15( rw): : -1
15: ipRouteNextHop.192.168.4.128.15( rw): 192.168.4.128
15: ipRouteType.192.168.4.128.15(-rw): indirect
15: ipRouteProto.192.168.4.128.15( ro): rip
15: ipRouteAge.192.168.4.128.15( rw): 4
15: ipRouteMask.192.168.4.128.15( rw): : 255.255.255.128
15: ipRouteMetric5.192.168.4.128.15( rw): -1
15: ipRouteInfo.192.168.4.128.15( ro): .:0.0
16: ipRouteDest.16.0.0.15.16( rw): 16.0.0.15
16: ipRouteIfIndex.16.0.0.15.16( rw): 1000
16: ipRouteMetric1.16.0.0.15.16( rw): 2
16: ipRouteMetric2.16.0.0.15.16( rw): -1
16: ipRouteMetric3.16.0.0.15.16( rw): -1
16: ipRouteMetric4.16.0.0.15.16( rw): -1
16: ipRouteNextHop.16.0.0.15.16( rw): 16.0.0.15
16: ipRouteType.16.0.0.15.16(-rw): indirect
16: ipRouteProto.16.0.0.15.16( ro): rip
16: ipRouteAge.16.0.0.15.16( rw): 5
16: ipRouteMask.16.0.0.15.16( rw): 255.255.255.0
16: ipRouteMetric5.16.0.0.15.16( rw): -1
16: ipRouteInfo.16.0.0.15.16( ro): .0.0
zeusbox:ipRouteTable>

```

1.2.5 External Commands

As listed by the [help](#) (?) command the following external commands are also available from the SNMP shell.

- [ping](#)
- [ipxping](#)
- [setup](#)
- [ifstat](#)
- [debug](#)
- [ospfmon](#)
- [telnet](#)
- [minipad](#)
- [update](#)
- [netstat](#)
- [date](#)
- [traceroute](#)
- [isdnlogin](#)
- [halt](#)
- [ifconfig](#)
- [modem](#)

The ping Command

Usage: **ping** [-i] [-f<precount>] [-d<msec>] [-c<count>] <target> [*<size>*]

The ping program can be used to test communication with another host. Ping sends ICMP echo_request packets of length *size* to *host*.

- -i: incremental. Each successive packet is sent with one additional byte.
- -f: flood. Each packet(s) is sent immediately after one is received. <precount> sets the number of packets to be sent without acknowledgement. The command -f 1 without -d nn sends/receives nearly half of the bandwidth.
- d: delay. The time in milliseconds to wait before the next packet is sent (default is 1000).
- -c: count. Only a specified number of packets is sent.
- Target is a required parameter which takes an IP address or a hostname.
- Size is optional and sets the length (in bytes) of the packets to use.

The ping command operates in continuous mode and keeps sending packets until the program is stopped by en-

tering Control-C; that is, holding down the "Ctrl" key ("Strg" key on German keyboards) and pressing "C".

The `telnet` Command

Usage: **telnet** *<host>* [*<port>*]

The `telnet` program can be used to establish a terminal session with the host specified by the *host* parameter. The host's numerical IP address or hostname can be used. The optional *port* parameter specifies which TCP port to connect to on the host.

The `traceroute` Command

Usage: **traceroute** [**-m** *<maxhops>*] [**-p** *<port>*] [**-q** *<nqueries>*] [**-w** *<waittime>*] *<address>* [*<packetsize>*]

By using the Internet Protocol's "Time-To-Live" field, the `traceroute` program prints the route packets take to arrive at a network host. The only mandatory parameter is the destination *address* which may be the host's name or numerical IP address. Options are used as follows:

- m** *<maxhops>* The maximum number of hops probe packets may travel before reaching *host* (i.e., the value of each packet's TTL, Time-To-Live field).
- p** *<port>* The UDP port to use. By default port 33333 is used. Traceroute requires that *host* is not using ports between *port* and $(port + maxhops - 1)$. If needed this option should be used to select an unused port range.
- q** *<nqueries>* The number of queries to send, default is 3.

- w** *<waittime>* Seconds to wait for a response to a probe packet.
- <address>* IP address (or hostname) of destination host.
- <packetsize>* The size (in bytes) to use for each probe packet.

The `ipxping` Command

Usage: **ipxping** [-c *<count>*] [-d *<delay>*] [-s *<net>*] [*<node>*]

The `ipxping` command can be used to test communication between the BinTec router and an IPX server and is comparable to IP's `ping` command. `ipxping` has one required argument, *net* which specifies the server's (or BinTec router's) IPX Network Number. The optional arguments are used as follows:

- c** *<count>* Send exactly *<count>* packets. By default one packet is sent (that is, if both **-c** and **-s** are not used).
- d** *<delay>* The time (in seconds) to wait between packets.
By default, 1 one second delay is used.
- s** Send 10000 packets.
- <node>* Optional IPX node number. Should be used if the IPX host's Internal Network Number is not = 0:0:0:0:1.

Note:



Even if the BinTec router's IPX network configuration is correct, the IPX server may not answer `ipxping` requests if it hasn't loaded its `IPXRTR.NLM`. It may be helpful to verify that the module is loaded if problems occur.

The `minipad` Command

Usage: **minipad** [-7] [-p <pktsz>] [-w <winsz>] [-c <cug>] [-o <outgocug>] [-b <bcug>] <x25address>

The `minipad` program is a basic PAD (Packet Assembler/Disassembler) program that can be used to provide remote login services for remote X.25 hosts. `Minipad` is also useful for testing X.25 routes. To disable incoming X.25 connections to `minipad`, set `x25LocalPadCall` to `dont_accept`.

`Minipad` has one mandatory argument, `x25address`, which can be a standard X.121 address, when preceded by an "@", or an extended X.25 address. Data calls to closed user groups defined in the `x25RouteTable` and `x25RewriteTable` can be placed using the `-c`, `-o`, and `-b` options.

<code>-7</code>	Use 7-bit data bytes.
<code>-p <pktsz></code>	The packet size to use.
<code>-w <winsz></code>	The window size to use.
<code>-c <cug></code>	Open connection with closed usergroup <code>cug</code> .
<code>-o <outgocug></code>	Open connection with outgoing closed user group <code>outgocug</code> .
<code>-b <bcug></code>	Open connection with bilateral closed user group <code>bcug</code> .
<code><x25address></code>	The remote host's X.25 address.

The `isdnlogin` Command

Usage: **isdnlogin** [-c <stknumber>] [-C] [-s <service>] [-a <addinfo>] [-b <bits>] <ISDN-number> <ISDN-servicename>

The `isdnlogin` program enables you to start a remote login shell on the BinTec router over ISDN. Using the `ISDN-number` and `ISDN-servicename` parameters, you select the

ISDN partner to login to, and the ISDN service to use. Valid *ISDN* servicenames are shown below.

Through D-channel signalling, *isdnlogin* can also accept incoming calls from analog modem with V.110 bitrate adaptation. Connections to V.110 stations can also be established with *isdnlogin* when the appropriate layer 1 protocol is supplied on the command line.

-c <stknumber>	The ISDN stack number to use.
-C	Attempt to use V42bis compression.
-s <service>	The 1TR6 service code to use for outgoing calls.
-a <addinfo>	The 1TR6 additional info code for outgoing calls.
-b <bits>	Use <bits>-bit data for transmission.
<ISDN-number>	The remote host's telephone number.
<ISDN-servicename>	The service names shown below are supported:

data	telephony	faxg3	faxg4
t_online	datex_j	btx	modem
56k	trans	dovb	
v110_1200	v110_2400	v110_4800	v110_9600
v110_14400	v110_19200	v110_38400	

The `setup` Command

Usage: **setup**

The `setup` command is used from the SNMP shell to start the Setup Tool program. Setup Tool provides a menu oriented interface to configuring the BinTec router, its major features, and administering/monitoring its operational state. The User's Guide is completely Setup Tool based;

please refer to it for information on using the Setup Tool program.

The `update` Command

Usage: **update** *<ipaddress>* *<filename>*

The `update` command can be used to upgrade the BinTec router's internal system software using TFTP. The host at *ipaddress* can be a UNIX system or a PC as long as it's been configured as a TFTP server. For PCs, DIME Tools includes a TFTP Server application. For UNIX systems see the section [Setting up a TFTP Server](#) in Chapter 5. The *filename* specifies the image to load into flash ROM. This image must be present in the TFTP-root directory configured on the server.

The `halt` Command

Usage: **halt**

The `halt` command halts the system and reboots using the default boot configuration file. The `halt` command has the same effect as powering the system off and on; i.e. it immediately shuts down the BinTec router. Therefore we recommend to better use `cmd=reboot`, because this way first running processes are completed, before the system is shut down (see [Rebooting the System](#)).

The `ifstat` Command

Usage: **ifstat** [-l] [-u] [*<ifcname>*]

The `ifstat` command displays status information for each of the system's interfaces, based on the contents of the *ifTable*.

`-l` Displays the full length of the interface descriptions (normally the description is

- limited to 12 characters).
- u Only display information for interfaces in the **up** state.
 - <ifcname> Only display information for interfaces whose description starts with the given characters (e.g. **ifstat en1** displays information on the interfaces en1, en1-llc, and en1-snap).

Status information for the desired interfaces is displayed in eleven columns as shown below.

Column	Meaning
ifTable Object	
Index	The BinTec router's interface number. Numbers > 10000 indicate software (or virtual) interfaces.
ifIndex	
Descr	The interface's description as assigned to ifDescr .
ifDescr	
Ty	Interface type. Integer values correspond to the enumerated types of the ifType object. 6=ethernet_csmacd, 7=iso88023_csmacd, 23=ppp, 32=frame_relay.
ifType	
Mtu	Maximum Transmission Unit for this interface; i.e., the largest network datagram that can be sent over this interface.
ifMtu	
Speed	The interface's estimated bandwidth in bits per second. For interfaces whose bandwidth doesn't change nominal bandwidth is reported.
ifSpeed	
St	The current operational status of the interface. May be: up (up), dn (down), ts (testing), do (dormant), or bl (blocked).
ifOperStatus	

Column	Meaning
<i>ifTable</i> Object	
Ipkts <i>ifInNUcastPkts</i> <i>ifInUcastPkts</i>	The number of packets received via this interface (the sum of <i>ifInNUcastPkts</i> and <i>ifInUcastPkts</i> objects) since <i>sysUpTime</i> .
les <i>ifInDiscards</i> <i>ifInErrors</i>	The number of incoming packets that couldn't be delivered due to errors (the sum of <i>ifInDiscards</i> and <i>ifInErrors</i> objects) since <i>sysUpTime</i> .
Opkts <i>ifOutNUcastPkts</i> <i>ifOutUcastPkts</i>	The number of packets requested via this interface (the sum of <i>ifOutNUcastPkts</i> and <i>ifOutUcastPkts</i> objects) since <i>sysUpTime</i> .
Oes <i>ifOutErrors</i> <i>ifOutDiscards</i>	The number of outgoing packets that couldn't be sent due to errors (the sum of <i>ifOutDiscards</i> and <i>ifOutErrors</i> objects) since <i>sysUpTime</i> .
PhysAddress/ ChgTime <i>ifPhysAddress</i> <i>ifLastChange</i>	The final column of the ifstat application display is divided into two parts: physical (MAC) address and change time. <ul style="list-style-type: none"> • On non-LAN interfaces, ifstat also shows the amount of time since the last change in the state of that interface. For example, if the interface is up, the length of time the interface has been up is displayed. • On LAN interfaces, on the other hand, the application will display the physical (MAC) address.

The netstat Command

Usage: **netstat** [-i] [-r] [-p]

The netstat command can be used to display a quick system status. Depending on which option is used, statistical information is retrieved from the *biboDialTable*, *ipxCircTable*, and *ipRouteTable*. The three options are as follows:

- i Display status information for each interface. Output is displayed in 10 columns similar to the ifstat command. See the description of [ifstat](#) for information on each column.
- p Display information for each configured ISDN partner. Output is displayed in seven columns as follows:

Column	Meaning
Index	Interface number taken from <i>biboPPPIfIndex</i> .
Partnername	The software interface's name as set in <i>IfDescr</i> .
Protocol	The protocol configured for this interface as set in <i>biboPPPEncapsulation</i> .
State	The current operational status as set in <i>ifOperStatus</i> .
Destination	Associates IP address with this partner. The displayed address' type is specified in the Type field.
Type	Type of address specified in the Destination field, may be: LOC (local), HOS (host), DEF (default), or NET (network).
Telno	Lists telephone numbers configured for the partner (<i>biboDialNumber</i>). A number's direction (<i>biboDialDirection</i>) is indicated by a greater-than sign (>) for outgoing, a less-than sign (<) for incoming, or both (< >) for both in and outgoing.

- r** Displays the current routing table entries. Output is displayed in seven columns as follows:
- e** Display extended routing parameters
- d** Display routes to destination.

Column	Meaning
Typ	Type of address specified in the Destination field, may be: LOC (local), HOS (host), DEF (default), or NET (network).
Destination	The destination IP address for this route as set in the <i>ipRouteDest</i> object.
Netmask	The netmask for this route as set in <i>ipRouteMask</i> .
Gateway	The IP address as set in <i>ipRouteNextHop</i> .
Met.	The current operational status as set in <i>ifOperStatus</i> .
Interface	The interface's name as set in <i>ifDescr</i>
Proto	Identifies how this route was learned as stored in <i>ipRouteProto</i> (local=manually configured routes).

The ifconfig Command

Usage: **ifconfig** <interface> [**destination** <destaddr>] [<address>] [**netmask** <mask>] [**up** | **down** | **dialup**] [-] [**metric** <n>]

The ifconfig command can be used to assign an address to a network interface and/or to configure network interface parameters and change the respective routing table entries.

When only the required interface parameter is used, ifconfig displays the current settings for the interface.

Options (and their respective *ipRouteTable* entries) are used as follows:

<i><interface></i>	Interface name (<i>ifDescr</i>)
destination <i><destaddr></i>	Destination IP address of a host for adding host routes. (<i>ipRouteDest</i> , <i>ipRouteMask</i>)
<i><address></i>	BinTec router's IP Address for this interface. (<i>ipRouteNextHop</i>)
netmask <i><mask></i>	Netmask of interface (<i>ipRouteMask</i>)
-	Don't define own IP address (i.e. <i>ipRouteNextHop</i> = 0.0.0.0)
metric <i><n></i>	Sets route metric to <i>n</i> (<i>ipRouteMetric1</i>)

The debug Command

Usage: **debug** [-t] [show | all | [*<subs>* [*<subs>* ...]]]

The debug command can be used to selectively display debugging information originating from one or more of the various subsystems. Command line parameters are used as follows:

-t	Print a timestamp before each debugging message.
show	Show all possible subsystems that can be debugged. ACCT ISDN INET X.25 IPX CAPI PPP BRIDGE CONFIG SNMP X.21 TOKEN ETHER RADIUS TAPI OSPF FR MODEM RIP
all	Display debugging information for all subsystems.

<subs> One or more subsystems separated by whitespace can be entered to display only debugging information from these subsystems. Current BinTec router subsystems include:

The date Command

Usage: **date** [-i] [YYMMDDHHMMSS]

The date command is used to set or display the current time. All BinTec router products have a software clock which stores the current time as retrieved from the host at *biboAdmTimeServer*. The optional date-string sets the current date to the specified Year, Month, Day, Hour, Minute, and Second.

Note that the BRICK-XM and BRICK-XL also have a real-time clock (hardware). The **-i** option displays the date stored in the software clock and is therefore only available on the XM and XL

Product	Date Command:	
	date	date <YYMMDDHHMMSS>
VICAS	Displays date currently stored in the <i>software</i> clock	Sets the <i>software</i> clock to <YYMMDDHHMMSS>
BRICK-XS		
BRICK-XM	Displays date currently stored in the <i>hardware</i> clock	Sets the hardware AND software clocks to <YYMMDDHHMMSS>
BRICK-XL		

The modem Command

Usage: **modem** [**update** <TFTP host> <TFTP filename> | **status**]

The modem command is used to update the system software of your CM-2XBRI module and FM-8MOD modem connector module, or to display the current operating status of all modems. Note that the FM-8MOD module is only available on the BRICK-XL. The command can be used as follows.

If the keyword **update** is used the following parameters are required.

<i><TFTP host></i>	The IP address of your TFTP server; i.e. the host where the modem software image can be retrieved.
<i><TFTP file></i>	The file name of the modem software image.

If you supplied the correct TFTP host and file name you will see some screen output concerning the loading and verifying of the image file. The update application will automatically detect all your modem connector modules and you will be queried to update each one individually.

If you reply with **y** the update will be performed. This will take approximately 60 seconds. After the modem update is complete you should reboot your BinTec router immediately if you want to use the new modem software.



Note To update the modem software image, a TFTP server (where your BinTec router can retrieve the software image) must be configured (see [Setting up a TFTP Server](#) in Chapter 5 for additional information).

When the **status** keyword is used, the system displays the current status for each modem similar to the following.

No	State	OBytes	IBytes	LastMessage
00	IDLE	280	2704	CONNECT 115200/K56/LAPM/NONE/38000:TX/31200:RX
01	IDLE	278	2701	CONNECT 115200/V34/LAPM/V42BIS/33600:TX/33600:RX
02	IDLE	18481	22233	CONNECT 115200/K56/LAPM/NONE/40000:TX/31200:RX
03	CALLING	0	0	
04	CONNECTED	59635	64330	CONNECT 115200/V34/LAPM/NONE/33600:TX/33600:RX

```

05  CONNECTED  407      79    CONNECT 115200/K56/LAPM/V42BIS/36000:TX/31200:RX
06  CALLED     0          0
07  IDLE       0          0

```

The `ospfmon` Command

Usage: `ospfmon db [rtr|net|sum|as-br|ext|stat] <options>`

The `ospfmon` application can be used from the SNMP shell to display the contents of the BinTec router's OSPF Link State Database. Note that only LSA header information is stored in the MIB system tables, this application can be used to dump the complete contents of the database. The various parameters can be used to selectively display specific types of database entries.

Only one of the six identifiers can be used at time to display a cross section of the database.

rtr	Show all Router links.
net	Show all Network links.
sum	Show all Summary links.
asbr	Show all AS Border Router links.
ext	Show all External Links.
stat	Show OSPF database statistics.

Additional options may also be used to further identify more specific types of entries and include.

area <id>	Show database entries for area <id>.
rtrid <id>	Show entries generated by router ID <id>.
lsid <id>	Show database entry with link state ID <id>.

Example:

Router Links from the Link State Database for Area 0.0.0.0 (from BRICK-XL in this diagram) might look like this.

```
BRICK-XL:> ospfmon db rtr area 11.0.0.0
Area 11.0.0.0

Router Link Age 920 Options 0x20 Lsld 192.168.30.1
Rtrld 192.168.30.1Seq 0x80000002 Checksum 0xe72a Len 48
options 0x2 links 2
Stub Network id 12.0.0.2 data 255.255.255.255 metric 1562
Stub Network id 12.0.0.3 data 255.255.255.255 metric 0
```

Note that the Link State ID (Lsld) of the database entry has different meanings based on the type of Link State Advertisement that is displayed. The table below shows the meanings for the five LSA types.

LSA Type:	Meaning of Link State ID:
Router Link	The Router ID of the router that generated the LSA.
Network Link	The IP Address of the DR on the destination network
Summary Link	The ipRouteDest of the propagated IP route.
ASBR Summary Link	The Router ID of the Autonomous System Border Router.
External Link	The ipRouteDest of the propagated IP route.

1.3 BinTec router System Tables

When booting the BinTec router loads its configuration file into memory. Normally the configuration file (named

"boot") is loaded from flash memory. (A configuration file can also be loaded from a remote TFTP host at any time during operation.)

The BinTec router's configuration file consists of system tables and variables whose format and structure are defined in [The MIB](#). Upon loading this information is stored in memory (RAM) and can be seen as a sort of relational database whose current contents can be manipulated from the SNMP shell. Each table in the database consists of rows and columns where:

- Column headings represent individual MIB object type.
- Rows consist of instances of several MIB objects.

There are static tables only containing just one row (e.g. *system*). Tables with multiple rows are numbered (*inx*) starting from 00., Thus each table entry, or row, refers to an instance of several MIB objects, or variables. The *ipNetToMediaTable* (the current ARP cache) is shown below.

<i>ipNetToMediaTable</i>				
<i>inx</i>	IfIndex (*rw)	PhysAddress (rw)	NetAddress (*rw)	Type (-rw)
00	1000	8:0:24:af:b2:3	192.168.6.140	static
01	1000	0:a0:f9:c7:4:4	192.168.6.12	static
02	2000	0:8:2:4b:4e:24	192.168.6.5	dynamic
03	2000	0:af:92:5a:1:2	192.168.6.37	dynamic

The characters (in parentheses) following each column name have special meanings for [creating](#) and [deleting](#) table entries. The *inx* number identifies a specific row and can be used when [editing](#) table entries.

*	Identifies index objects. Index objects define a unique database key that is required when creating new table entries.
-	Identifies the variable that contains the delete flag. These variables are used to delete a table entry.

ro	Identifies a variable as being Read-Only . These variables contain values that may not be changed.
rw	Identifies a variable as being Read-Write . Values for these variables can be changed.

1.3.1 Short vs. Long Names

When **Creating, Deleting, or Editing** BinTec router system table entries, MIB variables are normally identified from the command line using their complete (or **Long Name**) name as defined in [The MIB](#). Long Names for the MIB objects defined in the *ipNetToMediaTable* are:

```
ipNetToMediaIfIndex  
ipNetToMediaPhysAddress  
ipNetToMediaNetAddress  
ipNetToMediaType
```

Note that objects contained in the system table currently displayed in [The Shell Prompt](#) are also accessible via their **Short Names**. This allows the shell prompt to operate as a sort of *Current Working Directory*. Before changing (or creating) table entries from the SNMP shell, you'll probably want to display the table's contents first. As the table's contents are written to the screen the table's short names are displayed. The short names for MIB objects contained in the *ipNetToMediaTable* (shown on the previous page) are:

```
IfIndex  
PhysAddress  
NetAddress  
Type
```

Note:

MIB objects are NOT case sensitive. Upper and lower case characters have been used above for added readability. System table entries can be manipulated using any combination of upper/lower case characters with either long or short names as explained above.

1.3.2 Creating Table Entries

Creating table entries is comparable to adding a new entry into the database that's currently stored in memory.¹ Since variables in the database are individual instances of MIB objects each variable must be identified by a unique key. Each table row contains a unique database key which consists of the values of all the index objects for that row. And because a table row can only contain one instance for each MIB object, the key identifies the instances of all variables for the row.

In the *ipNetToMediaTable* shown above, the *IfIndex* and the *NetAddress* objects are index variables. The four instances of the *PhysAddress* object can be uniquely identified in the database their respective keys as follows:

<i>PhysAddress</i> Instance	DB Key (<i>IfIndex.NetAddress</i>)
8:0:24:af:b2:3	1000.192.168.6.140
0:a0:f9:c7:4:4	1000.192.168.6.12
0:8:2:4b:4e:24	2000.192.168.6.5
0:af:92:5a:1:2	2000.192.168.6.37

For index objects that are [Enumerated Types](#) the numeric value is always used. If the *IfIndex* object consisted of the values ethernet (1), token_ring (2), or other (3) the respective keys shown above might be: 1.192.168.6.140, 1.192.168.6.12, 2.192.168.6.5, and 2.192.168.6.37.

1.The reference to "memory" is here because changes to a table's contents are only saved to a writeable medium (flash ROM or a remote system's disk) upon explicit instruction.

This complicated explanation simply means that in order to create a new table entry, a new database key has to be defined which involves setting all index variables within one command. Additional variables may also be set at the same time. Variables not defined when a row is created are assigned default values that may be changed later (see [Editing Table Entries](#)).

A new static ARP mapping entry (comparable to: `arp -s` on most operating systems) could be added to the *ipNetToMediaTable* shown above using the following command.

```
mybrick:system> ipNetToMediaIfIndex=1000 ipNetToMediaNetAddress=192.168.6.6
ipNetToMediaPhysAddress=0:4:f1:c0:8:f3

mybrick:ipNetToMediaTable>
```

In this example, setting the *ipNetToMediaPhysAddress* object is not actually required for creating the table entry, however, it makes sense to associate the IP address with the hardware address within the same command.

Some system tables contain MIB objects for which only one instance is possible (the *admin* table for example; which among other things, contains the TCP port numbers the BinTec router uses). Logically these tables (called static tables) can only contain one row.

NOTE



Some system tables are not intended to be changed manually (e.g., tables that contain ISDN call or IP session logging information) and only contain Read-Only variables. Although MIB objects are marked as index variables (with an ***) in these tables they're also marked **ro**; meaning that only the BinTec router can update these tables.

1.3.3 Deleting Table Entries

To delete a table entry the variable (in the row that you want to delete) containing the delete flag must be set to de-

lete. The delete flag is denoted by the (-) character in parentheses in the column name.

As mentioned above the database key identifies all instances of MIB objects on a specific table row. This key could be used to identify a specific instance of the delete object thereby deleting a complete table row. The format is `<MIB object>:<DB Key>=delete`. The first row in our [ipNetToMediaTable](#) could be deleted in this manner using the command:

```
ipNetToMediaType.1000.192.168.6.140=delete
```

So that you don't have to decipher database keys (which can sometimes be long and consist of multiple variables) the best way to remove a table entry is to append the table row number to the delete object (separated by a colon). The format is `<MIB Object>:<inx number>=delete`. The same row could be deleted from our [ipNetToMediaTable](#) using:

```
ipNetToMediaType:0=delete
```

or

```
ipNetToMediaTable:0=delete
```

The row numbers of each table are indicated by the *inx* number which is shown when displaying a table's contents to the screen.

1.3.4 Editing Table Entries

The contents of a specific instance of a MIB object, i.e., the contents of a specific table cell, can be changed. Both methods mentioned in [Deleting Table Entries](#) can be used. Again the preferable (easier) method involves using the *inx* value to identify the table row.

We could change the hardware address associated with IP address 192.168.6.5 in our [ipNetToMediaTable](#) with either of the following commands.

```
ipNetToMediaPhysAddress.1000.192.168.6.5=0:0:f3:a0:3:f1
```

```
ipNetToMediaPhysAddress:0=0:0:f3:a0:3:f1
```

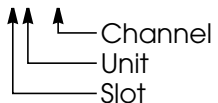
Of course variables can only be assigned values that are appropriate to the respective [Object Types](#).

1.4 BinTec router Interfaces

One of the key concepts used on the BinTec router is the idea of interfaces; however, several different types of interfaces are used. These include the following which are described below.

- Special Interfaces
- Hardware Interfaces (i.e., the physical interface)
- Software Interfaces (also referred to as *virtual* interfaces)

The numeric value of the *ifIndex* variable, used in many BinTec router system tables identifies a specific BinTec router interface. The *ifIndex* is a five digit number (leading 0s are normally not shown) that identifies the interface's type and some of the special characteristics of the interface which are described in the following sections.

Type	Range	Comments
Special Interfaces	0	The REFUSE Interface
	1	The LOCAL Interface
	2	The IGNORE Interface
Hardware Interfaces (Physical Interfaces)	1000	SUCC 
	...	
	9999	
Software Interfaces (Virtual Interfaces)	10000	Software interfaces sequentially ordered by category, see: (Software Interfaces)
	...	
	99999	

This shows the initial breakdown based on the interface types; software and hardware interfaces can be broken

down further according to their specific characteristics. This is explained in the following sections.

1.4.1 Special Interfaces

Three special (destination) interfaces are available on the BinTec router and are mainly useful when creating special routes for handling different situations depending on the characteristics of the interface.

These interfaces are always listed first in the *ifTable* (Interface Table) and have the following characteristics.

The REFUSE Interface (ifIndex = 0)

When packets are routed to the REFUSE interface (in the *ipRouteTable* and the *ipExtRtTable*) the packet is discarded and an "ICMP Destination unreachable" message is transmitted to the sender; i.e., the host at the address identified in the Source IP Address field of the IP datagram (see the diagram of the [Internet Layer](#) in Chapter 6).

The LOCAL Interface (ifIndex = 1)

Packets routed to the LOCAL interface are given to an appropriate internal process on the BinTec router such as the BinTec router's minipad application.

The IGNORE Interface (ifIndex 2)

Packets routed (via the *ipRouteTable* and the *ipExtRtTable*) to the IGNORE interface are discarded meaning that the packet is not forwarded. This destination interface is similar to the REFUSE interface with one exception. No status information is transmitted to the sender for packets routed to this interface.

1.4.2 Hardware Interfaces

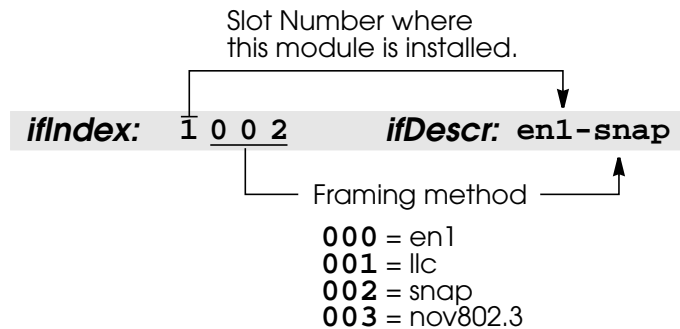
BinTec router hardware modules are listed by SBus slot in the *biboAdmBoardTable*. Both Feature Modules (FM) and Communications Modules (CM) are shown there. Communications modules that provide hardware interfaces routing capable of routing are listed in the *ifTable* and are identified by *ifIndex* values that are in the range:

$$\begin{array}{ccccc} \textit{ifIndex} & & \text{HW Interface} & & \textit{ifIndex} \\ 1000 & \leq & \text{value} & < & 10000 \end{array}$$

These interfaces consist of Point-To-Multipoint interfaces (such as ethernet, and token ring interfaces), and Point-To-Point interfaces (such as ISDN S₀, ISDN S_{2M}, and X.21 interfaces).

Point-to-Multipoint

Point-to-multipoint interfaces (Ethernet and Token-Ring) are listed in the *ifTable*. The value of the *ifIndex* and *ifDescr* objects are encoded as follows.



The *ifIndex* shown above identifies a point-to-multipoint interface installed in slot 1. On most systems this is the CM-BNCTP ethernet module but may also be a token ring mod-

ule (CM-TR). The *biboAdmBoardTable* entries would verify this module. This hardware interface uses SNAP framing. For information on the frame formats used with point-to-multipoint interfaces refer to [Appendix B](#).

Point-to-Point

Point-to-point interfaces include various forms of X.21, X.25, and ISDN interfaces. These different types of point-to-point interfaces depend on the type of installed hardware, how the hardware is configured, and where (which SBus slot) the hardware is installed.

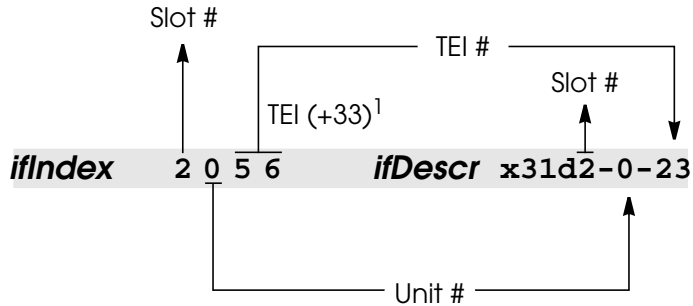
X.21 Interfaces

- X.21 interfaces are listed in the *ifTable*. Only the slot digit is used and identifies the applicable slot for the CM-X21 module (i.e., 3000 for CM-X21 in slot 3). A corresponding entry (*x21IfIndex*) is also present in the *x21IfTable*.

X.31 Interfaces

- X.31 (in the D-Channel) Interfaces
X.31 interfaces are listed in the *ifTable*. A corresponding entry is also found in the

x25LinkPresetTable. The *ifIndex* used for X.31 interfaces is encoded as follows.



1.) 0 - 32 are reserved for ISDN leased B-Channel interfaces

ISDN Interfaces

- **Dialup Interfaces**

ISDN Dialup interfaces are not listed in the *ifTable* since they do not provide directly routable interfaces; this is where the software interfaces are required.

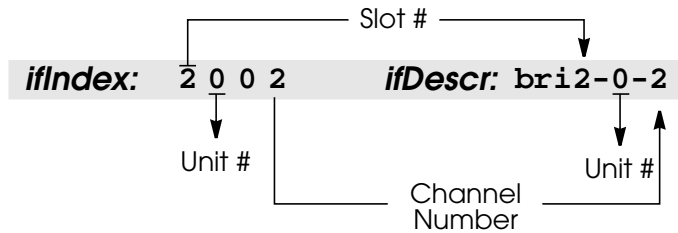
- **ISDN Leased Line Interfaces**

Leased line interfaces are listed in the *ifTable* since these interfaces identify a directly routable interface. Two types of leased line interfaces exist: interfaces consisting of a single ISDN B-channel (S_0) and interfaces that consist of multiple ISDN B-Channels (S_0 or S_{2M}), called Bundles.

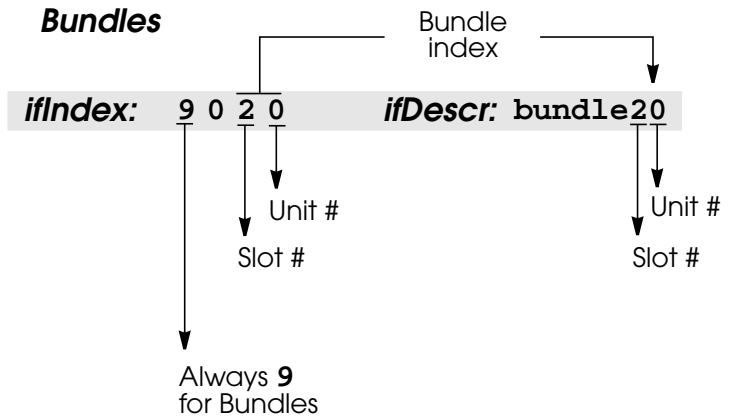
For Leased line interfaces consisting of a single B-Channel, the *ifIndex* and *ifDescr* objects are encoded

ed as follows:

Leased B-Channel



For bundle interfaces the *ifIndex* and *ifDescr* objects are encoded as follows:



1.4.3 Software Interfaces

Software interfaces are also referred to as *virtual* interfaces since they are mapped to one or more hardware interfaces (point-to-point or point-to-multipoint). The most common examples are dial-up ISDN partner interfaces. When setting up these software interfaces you may decide to associate

one or more ISDN B-channels from one or more ISDN ports (BRICK-XM and BRICK-XL only) that are used to accept calls from, or place calls to, the partner.

Software interfaces are listed in the *ifTable* and are identified by *ifIndex* values that are in the range:

$$\begin{array}{ccccc} \textit{ifIndex} & & \text{SW Interface} & & \textit{ifIndex} \\ 10001 & \leq & \text{value} & < & 29999 \end{array}$$

Software interfaces can be further distinguished as follows:

<i>ifIndex</i>	SW Interface Type
10001 ... 14999	Dial-Up ISDN Interfaces
15001 ... 15999	RADIUS Interfaces
18001 ... 19999	Frame Relay over ISDN Interfaces
20000 ... 29999	Multiprotocol over X.25 Interfaces

1.5 BinTec router Configuration Files

BinTec router configuration files are stored either locally on the BinTec router or on a remote host (TFTP server). When booting this information is loaded into memory and becomes the BinTec router's active configuration.

Configuration files stored locally on the BinTec router are stored in FLASH PROM (programmable read-only memory) memory which we simply refer to as flash. The contents of flash can be seen as a directory whose contents are listed in the *biboAdmConfigDirTable*. Configuration files stored in flash can be managed using the commands described in [Managing FLASH files](#) below.

Configuration files can be sent to or retrieved from remote hosts using the TFTP commands described in [Transferring Files with TFTP](#).

The system can also be rebooted using the `cmd=reboot` command described in [Rebooting the System](#).

1.5.1 Managing FLASH files

To help you manage different configuration files, the BinTec router uses the *biboAdmConfigTable*. This table contains the fields *Cmd*, *Object*, *Path*, *Pathnew*, *Host*, *State*, and *File*. This table is read by the configuration daemon, the `configd` process, which periodically:

1. Reads table entries.
2. Performs requested actions, using the respective field values as command parameters.
3. Updates the respective State field according to the State of the requested command.
4. Removes table entries once the respective action is performed.

An action is requested by assigning a value to the fields appropriate to the command, the **configd** process executes the requested actions.

The *State* field is updated intermittently while performing the action. The *State* field may be; *todo*, *running*, *done*, or *error*, depending on the status of the requested action. If the command resulted an error condition, you can find a detailed explanation of what caused the error by viewing the *biboAdmSyslogTable*. Once the requested action is completed the results can be seen by viewing the *biboAdmConfigDirTable*.

Tip

When using third party SNMP managers configuration data can be managed by accessing the respective objects in the *biboAdmConfigTable*.

Configuration files can be stored on the BinTec router (i.e., in flash) using the commands shown below. Although this information is presented in command syntax notation the actual commands simply involves assigning various parameter values to the contents of the *biboAdmConfigTable*.

```
cmd=save [path=<dirname>] [object=<tableobj>]  
cmd=load [path=<dirname>] [object=<tableobj>]  
cmd=delete path=<dirname> [object=<tableobj>]  
cmd=copy path=<oldname> pathnew=<newname>  
cmd=move path=<oldname> pathnew=<newname>
```

Saving Configuration Files

The BinTec router allows you to have multiple configuration files as long as there is enough room in flash to store them. To make sure that your configuration information (and any changes you have made while the system is running) is available after every system bootup, you must instruct the BinTec router to write the configuration data. This

is done by assigning the value "save" to the *biboAdmConfigCmd* field.

Usage:

```
cmd=save [path=<dirname>] [object=<tableobj>]
```

Optional arguments:

- path** Specifies the file in flash to write data to. The default value is "boot". If *dirname* contains spaces, the name must be enclosed in double quotes.
- object** Specifies the object(s) to save. Either a specific table of information can be saved or a complete configuration. If no objects are specified, all tables are written to path.

The result of this command is that all Read-Write information is written to the file specified by path. If the optional parameters are not used, the complete configuration (all tables) is saved to the "boot." file.

For example,

```
cmd=save
```

would write all configuration information to flash as "boot". If you want to save just the *ipRouteTable* in the file *test*, you could issue:

```
cmd=save path=test object=ipRouteTable
```

You can verify the actions have been completed by listing the entries in *biboAdmConfigDirTable*. You should see a listing of each configuration file you saved. Each line shows you the name of the file (*Name*), the number of tables saved in the file (*Count*), and the contents of the file (*Contents*), i.e., "all" or a list of individual table numbers separated by colons.

Note:

The internal procedure of writing configuration files can take between 5 and 20 seconds. It is strongly recommended that during this time no additional changes be made to the configuration. Verify your current changes before making additional ones.

Loading Configuration Files

During system initialization, the default configuration file ("boot") is loaded into memory. This boot file may be loaded locally or via a remote system, using BootP. The working state of the BinTec router is dependent upon on the configuration information in active memory. A new configuration file (or a single table) can be loaded into memory from flash while the system is running. This is done by assigning load to *biboAdmConfigCmd*.

Usage:

```
cmd=load [path=<dirname>] [object=<tableobj>]
```

Optional arguments:

- path** Specifies the file in flash to load data from. Default is "boot". If *dirname* contains spaces, the name must be enclosed in double quotes.
- object** Specifies the object(s) to load into memory. All tables can be loaded from a file or individual tables. If no objects are specified, all tables are loaded from path.

For example, to load a configuration from flash from the file "test"

```
cmd=load path=test
```

would be used; while the command

```
cmd=load path=test object=admin
```

would only load the *Admin* table from “test”. After loading configuration information, changes take effect automatically since the information is loaded directly into memory.

Deleting Configuration Files

Deleting complete configuration files or specific tables within them is done by assigning “delete” to the *Cmd* field.

Usage:

```
cmd=delete path=<dirname> [object=<tableobj>]
```

Required arguments:

path Specifies the file in flash RAM to remove data from.
Default is “boot”. If *dirname* contains spaces, the name must be enclosed in double quotes.

Optional arguments:

object Specifies the objects to remove. If no objects are specified, all tables are removed from *<path>*.

For example, to delete all configuration information from flash file “test”,

```
cmd=delete path=test
```

could be used; to delete only the *ipRouteTable* from the flash file “test” you could use

```
cmd=delete path=test object=ipRouteTable
```

Copying Configuration Files

To copy configuration files you can assign the value “copy” to the *biboAdmConfigCmd* object. When using “copy” the pa-

Tip

When deleting configuration files you may notice that the amount of available memory space shown in the **biboAdmConfigDirTable** is not adjusted. This is because flash can't be erased progressively; configuration files are only marked for deletion. When the flash becomes full, the system automatically reorganizes flash RAM, deleting previously marked data.

Note:

You can delete the contents of the flash RAM completely by assigning "/" to the path parameter. It is recommended however, that you save all configuration information to a remote host using TFTP and the [cmd=put](#) assignment before using this syntax.

Parameters differ slightly from their previously discussed usage.

Usage:

```
cmd=copy path=<oldname> pathnew=<newname>
```

Required arguments:

- path** Specifies the file in flash to copy data from. If *path* contains spaces, it must be enclosed in double quotes.
- pathnew** Specifies the new file in flash to write the data to. If *pathnew* contains spaces, it must be enclosed in double quotes.

This command is not capable of selecting individual tables from path; only complete files can be copied. Thus, the command:

```
cmd=copy path=boot pathnew=backup
```

copies the configuration file "boot" to the file "backup".

Moving Configuration Files

You can assign the value “move” to the *Cmd* field to rename configuration files. This has the same effect as issuing the [cmd=copy](#) and [cmd=delete](#) commands consecutively.

Usage:

```
cmd=move path=<oldname> pathnew=<newname>
```

Required arguments:

path Specifies the file in flash to remove.
pathnew Specifies the new file in flash to create.

The result of this operation is that the file <oldname> is renamed to <newname>. To rename the “boot” file to “old-boot” use the command:

```
cmd=move path=boot pathnew=oldboot
```

1.5.2 Transferring Files with TFTP

Using [TFTP \(Trivial File Transfer Protocol\)](#) you can transmit and retrieve configuration files to and from remote hosts on your network. This is made possible using three additional enumerated values for the *biboAdmConfigCmd* object: **put**, **get**, and **state**.

To exchange configuration files with remote hosts you must first set up a TFTP server on these hosts. Information on setting up a TFTP server on UNIX machines is provided in Chapter 5 in [Setting up a TFTP Server](#). A TFTP Server application for PCs is included with [BRICKware for Windows](#).

The commands used for exchanging configuration information among remote TFTP hosts are as follows.

```
cmd=put host=<a.b.c.d> [path=<flashname>]  
[file=<filename>] [object=<tableobj>]  
cmd=get host=<a.b.c.d> [path=<flashname>]  
[file=<filename>] [object=<tableobj>]
```

```
cmd=state host=<a.b.c.d> [file=<filename>] [object=<tableobj>]
```

Note:

Configuration files may contain the current passwords for the Read, Write and Admin Communities. If you use the [cmd=put](#) or [cmd=state](#) commands to transfer BinTec router configuration files to remote hosts, you should also control access to these files for security reasons.

Sending TFTP Files

Once TFTP is setup you can assign “put” to the *biboAdm-ConfigCmd* object to transmit configuration information stored in flash to a file on a remote host. Only Read-Write information is included in the file.

The TFTP file to be written on the remote host must already exist (and, for UNIX hosts, must be world writable) prior to executing the command.

If problems occur in connection with older BSD based TFTP servers see the [Special Note](#): in Chapter System Administration.

Usage:

```
cmd=put [host=<a.b.c.d>] [path=<flashname>]  
[file=<filename>] [object=<tableobj>]
```

Optional arguments:

- host** Specifies the IP address of the host to send information to. A hostname can also be used if it can be resolved via DNS. If not specified the address set in *biboAdm-NameServer* is used by default.
- path** Specifies the file in flash RAM to copy data from. If not specified the default flash file is “boot”.
- file** Specifies the TFTP file to create on the remote host.

The file name is relative to the TFTP-boot directory configured on the host. (The default is **C:\BRICK** for PCs running DIME Tools' TFTP Server; or the last field of the tftp entry in **/etc/inetd.conf** on UNIX systems.) The file name defaults to "brick.cf" if *<file>* is not specified.

object Specifies the table object(s) to send. Either a specific table, or a complete configuration file can be sent. If no objects are specified, all tables are sent by default.

To retrieve the *ifTable* from the flash file "temp" and store the information in file "file.cf" on the host at 192.168.3.4 this command would be used:

```
cmd=put host=192.168.3.4 path=temp
file=file.cf ⇒
object=ifTable
```

Retrieving TFTP Files

You can also retrieve configuration data from remote hosts by assigning "get" to the *biboAdmConfigCmd* object. Once the retrieved file (or table information) is written to flash, the information can then be loaded into memory with [cmd=load](#) for it to take effect.

Usage:

```
cmd=get [host=<a.b.c.d>] [path=<flashname>]
[file=<filename> object=<tableobj>]
```

Optional arguments:

host Specifies the IP address of the host to retrieve data from.
A hostname can also be used if it can be resolved via DNS.

- If not specified the address set in *biboAdm-NameServer* is used by default.
- path** Specifies the file in flash to write data to. If the file already exists in flash its contents are overwritten. The default flash name is "boot".
 - file** Specifies the file on the remote host to retrieve data from. If not specified the TFTP file named "brick.cf" is requested.
 - object** Specifies the object(s) to retrieve. Here, either a specific table or a complete configuration file can be retrieved. If not specified all system tables are retrieved.

For example, using the command

```
cmd=get host=192.168.3.4 path=file.cf
file=temp ⇒
object=ifTable
```

would retrieve the *ifTable* from the file *file.cf* on host 192.168.3.4 and save it in a flash file named "temp".

Transmitting State Information

The previously mentioned TFTP commands only send or retrieve variables with Read-Write status. They also send/retrieve information from files stored in flash. Using "cmd=state", you can save all configuration information currently in memory and send the data to a remote TFTP host. This information includes Read-Write AND Read-Only data such as status/accounting information.

The TFTP file to be written on the remote host must already exist (and, for UNIX hosts, must be world writable) prior to executing the command.

If problems occur in connection with older BSD based TFTP servers see the [Special Note](#): in Chapter 5.

Usage:

```
cmd=state [host=<a.b.c.d>] [path=<flashname>]
          [file=<filename>] [object=<tableobj>]
```

Optional parameters:

host	Specifies the IP address of the host to send data to. If not specified the current value of <i>biboAdmNameServer</i> is used by default.
file	Specifies the file name on the remote host to write data to (and is relative to the TFTP boot directory on that host). If not specified the default file name is "brick.st".
object	Specifies the table(s) to retrieve data from. If no objects are specified, the contents of all tables are sent.

For example, using

```
cmd=state host=1.2.3.4
file=brick1.st ⇨
object=system
```

would retrieve all data from the *system* table and places it in `"/tftpboot/brick1.st"` on host 1.2.3.4 (if present the file is overwritten).

Tip

If you need to contact BinTec support, it is recommended that you have a complete state file available. This would be done with the command shown below.

```
cmd=state host=<IP Address>
```

where `<IP Address>` identifies a TFTP host you have access to.

1.5.3 Transferring Files with XMODEM via Serial Port

It is possible to load and save configuration files via the serial interface using the protocol XMODEM. Therefore the variable *file* is assigned the value `xmodem` or `xmodem-1k`. `xmodem-1k` uses a packet size of 1024 byte (default: 128 byte) and in general reaches a higher throughput. The packet size is defined by the sender so that the value `xmodem-1k` only makes sense on the sending end; on the receiving end it is ignored.

To make use of this new feature you have to access your BinTec router from a computer via the serial port and a terminal program.

Getting the Configuration

```
cmd=get file=xmodem path=new_config
```

loads a file received via XMODEM with the name `new_config` into the flash ROM of the BinTec router.

After this command has been started the terminal program must be set to Send (Upload) and the transmission

protocol (XMODEM) as well as the source file name and location must be entered. For the time of the file transfer the console cannot be used.

Putting the Configuration

```
cmd=put file=xmodem path=boot
```

sends the BinTec router's flash ROM file boot via XMODEM.

After this command has been started the terminal program must be set to Receive (Download) and the transmission protocol (XMODEM) as well as the destination file name and location must be entered. For the time of the file transfer the console cannot be used.

Transmitting State Information

The previously mentioned commands only send or retrieve the configuration files containing variables with Read-Write status. They send/retrieve information from files stored in flash. Using "cmd=state" you can save all configuration information currently in memory. This information includes Read-Write AND Read-Only data such as status/accounting information.

```
cmd=state file=xmodem
```

Note:



If you use cmd=put or cmd=state to transfer BinTec router configuration files, you should also control access to these files for security reasons.

When nothing is specified the currently selected baud rate is used for the transfer. The transfer baud rate can be changed by adding `@baud` to the file variable, e.g.:

```
cmd=put file=xmodem@9600 path=boot
```

Possible baud rates are 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200. For transmitting data to the BinTec router (`cmd=get`) you should not select a rate higher than 9600. Selecting higher than default baud rates may result in transmission errors. There are no limitations for BIANCA/BRICK-XL/XL2.

In case of transmission errors a syslog is generated.

This feature can only be used via the SNMP shell, not via Setup Tool.

1.5.4 Rebooting the System

The system can also be rebooted via SNMP by assigning “reboot” to the *biboAdmConfigCmd* object. This can be used for example, to stop and restart the system remotely from a telnet, isdnlogin, or minipad session.

```
cmd=reboot
```

No additional parameters are required.