Platform Notes: The SunATM Driver Software

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900
U.S.A. 650-960-1300

Part No. 806-5625-10
October 2000, Revision A

Send comments about this document to: docfeedback@sun.com

Please
Recycle

Contents

Figures

Tables

Preface

Platform Notes: The SunATM Driver Software provides configuration instructions for the Sun™ Computer Systems Supplement software used with the SunATM™ PCI and SBus adapters. These instructions are designed for system administrators with experience configuring ATM networks.

Using UNIX Commands

This document may not contain information on basic UNIX® commands and procedures such as shutting down the system, booting the system, and configuring devices.

See one or more of the following for this information:

- *Solaris Handbook for Sun Peripherals*
- AnswerBook2™ online documentation for the Solaris™ software environment
- Other software documentation that you received with your system

Typographic Conventions

Typeface or Symbol	Meaning	Examples
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `% You have mail.`
`AaBbCc123`	What you type, when contrasted with on-screen computer output	`% su` `Password:`
AaBbCc123	Book titles, new words or terms, words to be emphasized	Read Chapter 6 in the *User's Guide*. These are called *class* options. You *must* be superuser to do this.
	Command-line variable; replace with a real name or value	To delete a file, type `rm` *filename*.

Shell Prompts

TABLE P-1 Shell Prompts

Shell	Prompt
C shell	*machine_name*%
C shell superuser	*machine_name*#
Bourne shell and Korn shell	$
Bourne shell and Korn shell superuser	#

Related Documentation

TABLE P-2 Related Documentation

Application	Title
Hardware Installation	*SunATM Installation and User's Guide*
Release Information	*SunATM Release Notes*

Ordering Sun Documentation

Fatbrain.com, an Internet professional bookstore, stocks select product documentation from Sun Microsystems, Inc.

For a list of documents and how to order them, visit the Sun Documentation Center on Fatbrain.com at:

`http://www1.fatbrain.com/documentation/sun`

Accessing Sun Documentation Online

The `docs.sun.com`SM web site enables you to access Sun technical documentation on the Web. You can browse the `docs.sun.com` archive or search for a specific book title or subject at:

`http://docs.sun.com`

Sun Welcomes Your Comments

We are interested in improving our documentation and welcome your comments and suggestions. You can email your comments to us at:

docfeedback@sun.com

Please include the part number (806-5625-05) of your document in the subject line of your email.

CHAPTER 1

Installing the SunATM Software

These Platform Notes include instructions for installing and configuring the SunATM™ 5.0 software used by the following SunATM adapters:

- SunATM/S 155 UTP5 adapter
- SunATM/S 155 MMF adapter
- SunATM/S 622 MMF adapter
- SunATM/P 155 UTP5 adapter
- SunATM/P 155 MMF adapter
- SunATM/P 622 MMF adapter

Note – All references to the SunATM/S adapter refer to SunATM/S 155 version 2.0 or SunATM/S 622 version 2.1, unless otherwise noted. All references to the SunATM/P adapter refer to SunATM/P 155 or SunATM/P 622 version 3.0, unless otherwise noted.

This chapter contains the following sections:

Caution – Although this chapter describes how to install the SunATM software on your system, you *must* first configure the software *before* you reboot your system. See Chapter 2 "Configuring the SunATM Interfaces"," for instructions on how to use the `atmadmin` program to configure the software.

Preparing for the SunATM Software Installation

Removing Previous Versions of the SunATM Software

Before installing the SunATM software, remove any previous version of the SunATM software from your system. If you attempt to add the new software packages over existing SunATM packages, the installation will fail.

▼ To Remove Older SunATM Software Packages

1. **Become superuser (root).**
2. **Use the** `pkginfo` **command to check for any installed SunATM software packages:**

```
# /usr/bin/pkginfo | grep SUNWatm
```

If you find any SunATM packages, you must remove them before installing the SunATM 5.0 software.

1. **Remove any existing SunATM software packages by using the** `pkgrm` **command:**

```
# /usr/sbin/pkgrm SUNWatm SUNWatma SUNWatmu
```

Removing Links to Older SunATM 1.0 Device Entries

When you remove older versions of the SunATM software with the `pkgrm` utility, certain device entries in the `/dev` and `/devices` directories will not be removed. If you are installing the SunATM 5.0 software packages on a system in which a SunATM 1.0 SBus adapter had been installed, remove these device entries. The SunATM 1.0 hardware used the `sa` driver (which is not supported in the SunATM 5.0 software), and previous versions of the SunATM software created the `/dev/sa` and `/devices/pseudo/clone@0:sa` device entries.

1. **To remove these SunATM 1.0 device entries, type:**

```
# rm /dev/sa
# rm /devices/pseudo/clone@0:sa
```

Install the SunATM Adapter

Before installing and configuring the SunATM software, you must first install the SunATM adapter into your system and connect the adapter to an active network using the appropriate cable.

Installing the SunATM Software

The table below describes the SunATM software packages.

TABLE 1-1 SunATM Software Packages

Package	Description
SUNWatm	Contains the device driver software
SUNWatmu	Contains the man pages and the files required to configure an ATM Simple Network Management Protocol (SNMP) management system
SUNWatma	Contains the SunATM interim Application Programmers' Interface (API) libraries and header files

Note – For basic ATM functionality, the SUNWatm package is the only required software package.

▼ To Install the SunATM Software from the *Sun Computer Systems Supplement* CD-ROM

1. **Install the SunATM software as described in the** *Solaris Sun Hardware Platform Guide* **that shipped with these Platform Notes.**

   The SunATM packages will be installed in the following directories:

- SunATM Device Drivers and Utilities (SUNWatm) go into: `/kernel/mod`, `/kernel/mod/sparcv9`, `/kernel/drv`, `/etc/init.d`, `/kernel/drv/sparcv9`, `/etc/opt/SUNWconn/atm`, `/etc/rc2.d`, and `/etc/opt/SUNWconn/bin`.
- SunATM Runtime Support Software (SUNWatmu) goes into `/opt/SUNWconn/atm` and `/opt/SUNWconn/man`.

Note – Man pages contained in the SUNWatmu package go into `/opt/SUNWconn/atm/man` and will have symbolic links in `/opt/SUNWconn/man`. (To view these man pages, add the `/opt/SUNWconn/man` directory to your system's `$MANPATH` environment variable.) Interim API examples will go into `/opt/SUNWconn/atm/examples`.

- SunATM Interim API (`SUNWatma`) will go into `/opt/SUNWconn/atm/include`, `/opt/SUNWconn/atm/lib`, `/opt/SUNWconn/include` and `/opt/SUNWconn/lib`.

Checking the Package Installation Using `pkgchk`

Once the package is installed, you can use the `pkgchk` command to see if the installation was complete:

```
# /usr/sbin/pkgchk SUNWatm
```

You can specify multiple packages at the command line by separating the package names with a space. If you do not specify a package identifier, the entire contents of the machine are checked.

Checking the Package Installation Using `pkginfo`

Check the SunATM software installation by using the `pkginfo` command:

```
# /usr/bin/pkginfo | grep SUNWatm
system       SUNWatm   SunATM Device Drivers
application  SUNWatma  SunATM Interim Api Support Software
application  SUNWatmu  SunATM Runtime Support Software
```

Removing the Software Packages Using pkgrm

If you are superuser, you can remove one or more packages with the following command:

```
# /usr/sbin/pkgrm SUNWatm SUNWatma SUNWatmu
```

In this example, the pkgrm command will remove the SUNWatm (SunATM Device Drivers and Utilities), SUNWatma (SunATM Interim API Support Software), and SUNWatmu (SunATM Runtime Support Software) packages.

Configuring the SunATM Interfaces

After installing the SunATM software, you *must* configure the SunATM interfaces before you reboot your system. You can either use the SunATM configuration program, atmadmin, to configure the interfaces, or you can edit the SunATM configuration files directly.

See Chapter 2 “Configuring the SunATM Interfaces”” for instructions on how to use the atmadmin program, and see Chapter 3 “Editing SunATM Configuration Files”” for information about the SunATM configuration files.

Known Issues About the SunATM 5.0 Release

This section contains important information and news about the SunATM 5.0 software release.

Redundant LANE Servers

SunATM adapter software does not support redundant LAN Emulation Services, such as Cisco System’s Simple Server Redundancy Protocol (SSRP) for LAN Emulation.

SunATM and Solstice FireWall-1

The Solstice FireWall-1™ product does not currently support the SunATM 5.0 release. Refer to the bug report (BugID 4073989) for more information.

*hyper*SPARC Modules Are Not Supported

The SunATM 5.0 software does not support sun4m systems containing *hyper*SPARC™ modules. If you have a SPARCstation 10, 10SX, or 20 system with *hyper*SPARC modules, or if you have updated your system to use *hyper*SPARC modules, you will not be able to use this SunATM release.

To see if your machine contains *hyper*SPARC modules, type the following command:

```
# prtconf | grep Ross,RT625
```

You will see `Ross,RT625 (driver not attached)` as the result of this command if your system contains a *hyper*SPARC module.

CHAPTER 2

Configuring the SunATM Interfaces

This chapter describes the new features in the SunATM software and how to configure the software using the `atmadmin` configuration program.

Using the `atmadmin` Configuration Program

The SunATM configuration program, `atmadmin`, is an interactive command-line interface. The program contains a hierarchy of menus, which divide the configuration into six main parameter groups: system, physical layer, signalling, ILMI, Classical IP, and LAN Emulation. All but the system parameter group are specific to individual SunATM interfaces, so you must configure the parameters in these groups separately for each interface.

If you prefer, you can enter and change the SunATM configuration information by editing the SunATM configuration files directly. See Chapter 3 “Editing SunATM Configuration Files”,” for a description of the configuration files’ contents and formats.

Note – See the Glossary for descriptions of the ATM and SunATM terms used in this chapter. Chapter 3 “Editing SunATM Configuration Files”,” and Chapter 5 “Classical IP and LAN Emulation Protocols”,” also provide more information about ATM protocols and the SunATM implementation of these protocols.

Starting the atmadmin Configuration Program

The atmadmin program is installed with the SUNWatm software package in the /etc/opt/SUNWconn/bin directory. The program must be run as superuser (root). It can be run in any local or remote shell on the SunATM system.

```
# /etc/opt/SUNWconn/bin/atmadmin
```

atmadmin Main Menu

After you start the atmadmin configuration program, you see the atmadmin Main Menu. From this menu you can either go to the system parameter group menu (see "System Parameter Group Menu" on page 9), or you can enter the SunATM interface you want to configure. The following screen example is from a system with one interface named ba0.

```
Welcome to the SunATM Admin Program.
The following interfaces are installed in your system:
        ba0
        [S] Modify System Parameters
        [X] Exit
        [?] Help

Enter interface name or option: ba0
```

After selecting an interface, you will then see the Interface Configuration menu (see "Interface Configuration Menu" on page 10).

atmadmin Navigation Commands

TABLE 2-1 lists the basic commands that let you navigate through the menu hierarchy.

TABLE 2-1 Basic Navigational Commands in atmadmin

Command	Action
m	Return to the atmadmin main menu

TABLE 2-1 Basic Navigational Commands in `atmadmin` *(Continued)*

Command	Action
p	Return to the previous menu
x	Exit `atmadmin`
?	Provide more information about the options on this menu

System Parameter Group Menu

The system parameter group contains parameters that are not specific to an interface; they apply to the entire system. The following example shows the system parameter group menu.

```
Modifying system-wide parameters;
Currently configured as an ATM SNMP agent, using UDP port 1000
The SNMP agent options are:
        [A] ATM SNMP agent
        [N] not an agent
        [U] UDP Port


        [P] Previous Menu
        [M] Main Menu
        [X] Exit
        [?] Help

Enter selection:
```

ATM SNMP Agent Status

You can configure your SunATM system as an ATM SNMP agent. The SunATM SNMP daemon, `atmsnmpd`, always runs on an ATM host. If you do not run your system as an SNMP agent, the daemon does not bind to a UDP port.

Interface Configuration Menu

Once you select a SunATM interface, you will see the `atmadmin` Interface Configuration menu. From this menu you can proceed to the interface parameter group sub-menus, which are described in “`atmadmin` Parameter Groups” on page 11. You can use these sub-menus to change the SunATM interface configuration parameters.

```
Modifying ba0
        [Y] Physical Layer
        [U] UNI Signalling
        [I] ILMI Address Registration
        [C] Classical IP
        [L] LAN Emulation

        [P] Previous Menu
        [M] Main Menu
        [X] Exit
        [?] Help

Enter selection:
```

`atmadmin` and the SunATM Configuration Files in the `/etc/opt/SUNWconn/atm` directory

The `atmadmin` program first attempts to read the current configuration information from the `/etc/opt/SUNWconn/atm/atmconfig`, `/etc/opt/SUNWconn/atm/aarconfig`, and `/etc/opt/SUNWconn/atm/laneconfig` files. If no configuration information is found, or if the files do not exist, the default values listed in TABLE 2-2 are applied to the installed interfaces.

Caution – When saving configuration information, `atmadmin` *overwrites* the existing SunATM configuration files in the `/etc/opt/SUNWconn/atm` directory. Therefore, any comments or other changes you manually made to the files will be lost.

`atmadmin` Parameter Groups

The `atmadmin` configuration program contains a series of menus where you can input or alter the configuration of specific SunATM software parameters. These menus, or parameter groups, are described in this chapter:

TABLE 2-2 summarizes the configurable parameters in each parameter group. Although the parameter list appears rather lengthy, you only need to use the default values for most standard configurations. The large number of parameters offer the flexibility to support special case configurations, and to allow interoperability with equipment from other vendors.

Note – In most cases, you will only need to configure the parameters that do not have default values.

TABLE 2-2 Configurable Parameters in the SunATM Software

Group	Parameters	Possible Values	Default Values	Required?
System	SNMP Agent Status	`agent` or `not_agent`	`not_agent`	Yes
	SNMP Agent UDP port	0<= *n* <= 65355	1000	For SNMP Agent
Physical Layer	Framing Interface	SONET or SDH	SONET	Yes
Signalling	UNI Version	3.0, 3.1, 4.0, or none	No default	Yes
ILMI	ILMI Status	Enabled or Disabled	Enabled	Yes
Classical IP	Hostname/IP Address	Valid hostname and IP address	No default	For Classical IP
	Interface Type	Client, Server, or Standalone	No default	For Classical IP
	Local ATM Address	Valid ATM address	`$myaddress`	For Classical IP Clients or Servers

TABLE 2-2 Configurable Parameters in the SunATM Software *(Continued)*

Group	Parameters	Possible Values	Default Values	Required?
	ARP Server	Valid ATM address	`$localswitch_server`	For Classical IP Clients
	PVC	32 <= *n* < 1024	32	For Classical IP Standalones
	Destination hostname or IP address	Valid hostname and IP address	No default	For Classical IP Standalones
LAN Emulation	Instance Number	0 <= *n* <= 999	No default	For LAN Emulation
Per-Instance Parameters	Hostname/IP Address	Valid hostname and IP address	No default	For LAN Emulation
	Local ATM Address	Valid ATM address	`$myaddress`	For LAN Emulation
	LECS Indicator	`No LECS` or `LECS Present`	`LECS Present`	For LAN Emulation
	LECS ATM Address	Valid ATM address	ILMI value or the well-known LECS address	For LAN Emulation, `lecs_present`
	LES ATM Address	Valid ATM address	No default	For LAN Emulation, `no_lecs`
	Emulated LAN Name	Character string	No default	For additional instance on a physical interface
	Additional Hostnames?	Yes or no	No	For LAN Emulation
Per-Additional Hostname	Minor Instance Number	0 <= *n* <= 8190	None	For LAN Emulation, additional IP
	Hostname/IP Address	Valid hostname and IP address	No default	For LAN Emulation, additional IP

Physical Layer Parameter Group

The physical layer parameter group contains only the framing interface parameter. The following example shows the physical layer parameter menu.

```
Modifying ba0; Current framing interface is SONET
The framing interfaces that may be configured are:
        sonet
        sdh

        [P] Previous Menu
        [M] Main Menu
        [X] Exit
        [?] Help

Enter selection:
```

Framing Interface

The framing interface defines the encapsulation method used for ATM cells as they are sent onto the wire. The default framing interface is SONET, but the SunATM software also supports the SDH interface. Your switch product information should indicate whether your switch uses either the SONET or the SDH interface. If the switch uses the SDH interface, you will need to select SDH from the physical parameter group menu.

Signalling Parameter Group

The signalling parameter group contains only the UNI version parameter. The following example shows the signalling parameter menu.

```
Modifying ba0; Current UNI Version is 3.0
The UNI versions that may be configured are:
        3.0
        3.1
        4.0
        [N] No Signalling Enabled

        [P] Previous Menu
        [M] Main Menu
        [X] Exit
        [?] Help

Enter selection:
```

UNI Version

The SunATM software supports three versions of the ATM Forum's User Network Interface (UNI) Specification: versions 3.0, 3.1, and 4.0. You may choose not to enable signalling, but in order to support either Classical IP or LAN Emulation (or both), you must select one of the three UNI versions.

ILMI Parameter Group

If your ATM switch does not support the Interim Local Management Interface (ILMI), you can turn off the ILMI address registration on your SunATM interface from the ILMI configuration menu. The following example shows the ILMI configuration menu.

```
Modifying ba0; Currently ILMI is enabled
        [E] Enable ILMI
        [D] Disable ILMI

        [P] Previous Menu
        [M] Main Menu
        [X] Exit
        [?] Help

Enter selection:
```

Classical IP Parameter Group

Classical Internet Protocol (Classical IP), specified by RFC 1577, is one way of supporting the TCP/IP and UDP/IP protocols in an ATM environment. In Classical IP, an ATM ARP server is used to resolve IP addresses to ATM addresses, replacing the traditional ARP protocol. In this configuration, each host must register with the ARP server when the ATM interface is brought up. For more information on the Classical IP protocols, see “Classical Internet Protocol” on page 47.

One reason ATM ARP is used instead of the traditional ARP is that ATM does not support broadcast (a network capability providing transmission from one point to all points on a network). Because Classical IP over ATM does not support broadcast, you cannot use the `ypbind -broadcast` UNIX command to automatically locate the NIS server (ypserver) on a Classical IP ATM subnet.

If you are planning to run NIS over your ATM network, you must specify the list of NIS servers (ypservers) using the `ypinit -c` command. See the `ypinit(1M)` man page for details of setting up the ypserver. Be sure that the IP addresses of the ypservers are listed in the `/etc/hosts` file.

The Routing Information Protocol (RIP) also uses the broadcast feature of IP, so it is not supported under the Classical IP environment. In the Solaris operating environment, RIP is implemented by the daemon `in.routed`.

Classical IP alone also does not support the multicast packet delivery system. If you are using Classical IP, you must explicitly add the routes to the routers in the ATM subnet. You may also specify one router as the default router to provide connectivity outside of the ATM subnet. See the `route(1M)` man page for information on using the route command to add specific router entries and to add a default router.

You can use the Classical IP parameter group menu to define the Classical IP configuration of a SunATM interface.

```
Modifying ba0; Current Configuration:
    Arp Client
    IP = atm_cip
    ATM = $myaddress
    ARPSRV = $localswitch_server
        [N] No Classical IP Enabled
        [C] Client
        [S] Arp Server
        [T] Standalone
        [I] Hostname or IP Address
        [L] Local ATM Address
        [A] ATM ARP Server Address

        [P] Previous Menu
        [M] Main Menu
        [X] Exit
        [?] Help

Enter Selection:
```

Classical IP Interface Type

The SunATM software allows you to configure your interface as either a Classical IP ARP server or a client. In addition, you can connect two systems back-to-back, in a standalone configuration, using a Permanent Virtual Circuit (PVC). These three modes are options on the Classical IP parameter menu.

Hostname and IP Address

Regardless of the Classical IP Interface Type, you must assign an IP address and hostname to the interface. If you enter a hostname that appears in the `/etc/hosts` file, or if NIS, NIS+, or DNS is enabled and the hostname is resolvable over it, you are not prompted to enter an IP address. Instead, the resolution is performed

automatically. If the hostname cannot be resolved, you are prompted to enter an IP address. If you must enter an IP address, or if the address is only available through NIS, NIS+, or DNS, the SunATM software updates the `/etc/hosts` file.

A valid IP hostname is no more than 80 characters. A valid IP address is a set of four decimal numbers in the range of 0 to 255, separated by dots (for example, `149.144.130.9`).

Local ATM Address

The local ATM address is the 20-byte ATM address associated with a specific Classical IP instance. You must assign an ATM address to each Classical IP client and server, but you do not need to assign an ATM address on standalone (back-to-back) configurations. The following section describes ATM address formats and some of the SunATM software defined address variables.

ATM Address Formats and Variables

ATM addresses, like Network Service Access Point (NSAP) addresses, are 20 octets long, with each octet made up of 1 or 2 hexadecimal digits. The ATM address is divided into three fields: the End System Identifier field, the Selector field, and the Network Prefix field. The End System Identifier (ESI) field is a unique 6 octet value, which can be the IEEE hardware MAC address conventionally associated with every network interface. The Selector field is one octet long. The 13 octets that make up the rest of the ATM address are called the Network Prefix. This field should be derived from the ATM switch fabric to which the interface is connected. Every ATM switch fabric is configured with a 13 octet prefix.

To simplify references to ATM addresses in the SunATM software, several system-defined variables are built into the software. Variables are referenced with the `$` operator, as in UNIX shell scripts. TABLE 2-3 summarizes the system-defined SunATM ATM address variables.

TABLE 2-3 Predefined SunATM Variables

Variable	Description
`prefix`	The 13-byte prefix associated with the local switch.
`mac`	The 6-byte medium access control (MAC) address associated with the local host or interface.
`sel`	The default 1-byte selector for the local interface.
`macsel`	The concatenation of `$mac:$sel`.

TABLE 2-3 Predefined SunATM Variables *(Continued)*

Variable	Description
`myaddress`	Concatenation of `$prefix:$mac:$sel`, resulting in the default address for the local interface.
`sunmacsel`*N*	Concatenation of one of a series of reserved MAC addresses and `$sel` to create a block of reserved ATM ARP server addresses. *N* should be a decimal number in the range 0 - 199.
`localswitch_server`	Concatenation of `$prefix`, a unique reserved MAC address, and `$sel`. When used as a server address, restricts server access to clients connected to the local switch only.

Note – The `$prefix` variable, and any other variables that use it (including `$myaddress` and `$localswitch_server`), may not be used on interfaces that are not running ILMI.

ATM addresses are represented by 20 colon-separated octets, with each octet made up of 1 or 2 hexadecimal digits. You can combine variables representing portions of an ATM address with other variables and/or octets to make up a complete address. For example, `$prefix:aa:bb:cc:dd:ee:ff:$sel` represents a valid ATM address.

ATM ARP Server Address

If you configured the Classical IP instance as a client, you must also enter the address of the ARP server. This parameter, like the local ATM address, must be a 20-byte ATM address. See “ATM Address Formats and Variables” on page 17 for a discussion of ATM address formats and variables.

Permanent Virtual Circuit (PVC)

The Permanent Virtual Circuit parameter applies only to standalone configurations. It identifies the PVC which will be used to communicate between the two systems connected back to back. Both systems must use the same PVC value. The PVC parameter must be an integer (not hexadecimal) between 32 and 1023.

LAN Emulation Parameter Group

LAN Emulation, standardized by the ATM Forum's LAN Emulation 1.0 specification, is another way of providing TCP/IP and UDP/IP support over an ATM interface. Address resolution information is provided by a series of LAN Emulation services. When a LAN Emulation interface is brought up, it must register with these LAN Emulation services (known as "joining the LAN"). This registration process and the address resolution process are described in "LAN Emulation" on page 49.

Unlike Classical IP, the LAN Emulation protocol provides a broadcast service to the upper layer protocols. Therefore, the multicast and RIP limitations described in "Classical IP Parameter Group" on page 15, do not affect LAN Emulation interfaces.

The SunATM software allows a single ATM interface to join up to sixteen emulated local area networks (ELANs), provided that this action is allowed by the switch and LAN Emulation (LANE) services. Each ELAN joined is represented by a unique lane instance (for example, `lane0` or `lane1`).

Note – A requirement for supporting this feature is that the adapter card be assigned multiple MAC addresses, which the SunATM/S 2.1 and SunATM/P 3.0 adapters support. This feature *does not* work with the older SunATM/S 2.0 adapters. Use the `atmgetmac(1M)` command with the `count` option to find the number of MAC addresses assigned to your SunATM adapter.

After you configure LAN Emulation parameters, you are asked to choose an existing (previously configured) LAN Emulation (`lane`) instance or to create a new one in the LAN Emulation Instance menu. The following is an example of this menu.

```
The following lane instances are configured on ba0:
        lane0
        lane1
        [C] Create new lane instance
        [D] Delete lane instance

        [P] Previous Menu
        [M] Main Menu
        [X] Exit
        [?] Help

Enter lane instance or option: lane0
```

Per-Instance LAN Emulation

The Per-Instance LAN Emulation Parameters menu allows you to configure the per-instance LAN Emulation parameters.

```
Modifying lane0; Current Configuration:
    IP = atm_lane
    ATM = $myaddress
    LECS Present
    LECS_Address = well-known address
    no additional IP hostnames
        [I] Hostname or IP Address
        [L] Local ATM Address
        [C] LECS Present
        [N] No LECS
        [A] LECS ATM Address
        [E] Emulated LAN Name
        [H] Additional Hostnames

        [P] Previous Menu
        [M] Main Menu
        [X] Exit
        [?] Help

Enter Selection:
```

Hostname or IP Address

If IP traffic runs over a LAN Emulation instance, assign a hostname and corresponding IP address to the instance. If you enter a hostname that appears in the `/etc/hosts` file, or if NIS, NIS+, or DNS is enabled and the hostname is resolvable over it, you are not prompted to enter an IP address. Instead, the resolution is performed automatically. If the hostname cannot be resolved, you are prompted to enter an IP address. If you must enter an IP address, or if the address is only available through NIS, NIS+, or DNS, the SunATM software updates the `/etc/hosts` file.

A valid hostname is no more than 80 characters. A valid IP address is a set of four decimal numbers in the range of 0 to 255, separated by dots (for example, `149.144.130.9`).

Local ATM Address

The local ATM address is the 20-byte ATM address associated with this LAN Emulation instance. See “ATM Address Formats and Variables” on page 17 for more information about ATM address formats and variables.

Each `lane` instance must be assigned a unique ATM address. Each SunATM 2.1 or 3.0 adapter has been assigned 16 unique MAC addresses; if you use the variable `$myaddress` for each `lane` instance, the SunATM software will automatically distribute those MAC addresses to the `lane` instances associated with each physical interface.

LECS Indicator

Most LAN Emulation Services include a LAN Emulation Configuration Server (LECS), which is the first server contacted when bringing up a LAN Emulation client. The LECS provides the ATM address of the LAN Emulation Server (LES), as well as other configuration information about the emulated LAN. However, some LAN Emulation services do not include an LECS, and the LES must be contacted directly. With the LECS Indicator parameter, you specify which service should be contacted first in your configuration. The possible values for this parameter are displayed as individual options on the LAN Emulation Instance menu.

Note – If the value of this parameter is `No LECS`, you must specify a value for the LES ATM Address parameter.

LECS ATM Address

By default, the SunATM software attempts to obtain the LECS address using ILMI, as specified in the LAN Emulation specification. If this is not successful, the “well-known” ATM address, also specified by the ATM Forum, is used.

If your LECS uses a different ATM address (not the well-known address), and does not make that address available via ILMI, specify it using this parameter. If applicable, any of the ATM address variables described in “ATM Address Formats and Variables” on page 17 may be used. Use variable `$prefix`, in particular.

LES ATM Address

This parameter is required if the value of the LECS Indicator parameter is `no_LECS`. There is no “well-known” address for the LES, so an ATM address must be specified for the LES since there is not an LECS present to provide one. This parameter is a

standard ATM address. If any of the SunATM address variables described in Section 5.3.4.3 under “ATM Address Formats and Variables” (`$prefix` in particular) are applicable, they can be used.

Emulated LAN Name

If multiple Emulated LANs (ELANs) are present, you can enter a character string in the Emulated LAN Name parameter. The LAN Emulation client uses this parameter to tell the LAN Emulation services which ELAN it wishes to join. By default, if a SunATM LAN Emulation client does not specify an ELAN name, it tells the services to assign it to the default (or only) ELAN.

Note – If you have multiple LAN Emulation instances configured on a physical interface, only one instance can join the default (unspecified) ELAN. You must specify an ELAN name for all other instances.

Additional Hostnames

The SunATM software supports logical interfaces in the SunATM LAN Emulation environment. Logical interfaces allow you to assign multiple IP addresses to a single LAN Emulation interface. A logical interface name consists of three parts: the device name (in the case of SunATM LAN Emulation, `lane`); the major number, which corresponds to the `lane` instance number; and the minor number, which distinguishes the logical interfaces on a single `lane` instance. The format of a LAN Emulation logical interface name is `lane`*N*:*X*, where *N* is the major number and *X* is the minor number (for example, `lane0:2`).

The SunATM software associates each logical interface with a unique hostname and IP address. All logical interfaces on a given physical interface are associated with the same ATM and MAC addresses.

The hostname displayed in the LAN Emulation instance menu corresponds to the minor instance 0. The additional IP Address parameter indicates if any additional hostnames are assigned to the instance. Select this parameter to modify or create additional hostnames. You must enter or modify each additional IP hostname in the same manner as other IP hostname and address pairs (see “ATM Address Formats and Variables” on page 17 for more details), and associate it with a minor number between 0 and 255.

CHAPTER 3

Editing SunATM Configuration Files

This chapter describes how to configure SunATM interfaces by editing the configuration files.

You are not required to edit these configuration files by hand. You can use the `atmadmin` configuration program, described in “Using the atmadmin Configuration Program” on page 7, to configure the SunATM files. From the program’s command-line interface, you can change most of the SunATM parameters.

Caution – When it saves configuration information, `atmadmin` *overwrites* the existing SunATM configuration files in the `/etc/opt/SUNWconn/atm/` directory. Therefore, any comments or other changes you manually made to the files will be lost.

Editing the `/etc/opt/SUNWconn/atm/atmconfig` File

The `/etc/opt/SUNWconn/atm/atmconfig` file is a generic file that must appear on every SunATM system. It provides general configuration information used by the SunATM setup utilities to bring up ATM interfaces at boot time.

The file consists of one or more entries per interface. An entry contains the following fields described in TABLE 3-1:

TABLE 3-1 `/etc/opt/SUNWconn/atm/atmconfig` Field Descriptions

Field	Description
`Interface`	Physical interface, `ba`*N*.
`UNI/Framing`	Version of the UNI specification used for signalling, 3.0, 3.1, or 4.0; or, for entries with only two fields, the Framing Interface, SONET or SDH.
`CIP_Host`	Hostname used for Classical IP.
`LANE_Instance`	Instance number for a LAN Emulation interface; LAN Emulation interfaces will be called `lane`*N* where *N* is the LAN Emulation instance number. The LANE instance number must be between 0 and 999. Note: The LANE instance number is not necessarily the same as the physical instance number.
`LANE_Host`	Hostname used for LAN Emulation

The Interface and UNI fields are required for all interfaces. The CIP_Host field is required for interfaces that run Classical IP, and the LANE_Instance and LANE_Host fields are required for interfaces that run LAN Emulation. If a field is not used, it is represented by a hyphen.

Modifications to individual interfaces will take effect when the interface is plumbed. This will happen either at boot time or when you use the `atmifconfig` utility to plumb interfaces. If you modify an existing (already running) interface, you must first unplumb it with the `atmifconfig` utility. Refer to the `atmifconfig(1m)` man page or “Plumbing and Unplumbing Individual ATM Interfaces” on page 42 for more information.

Changing the Framing Interface in the `/etc/opt/SUNWconn/atm/atmconfig` File

The framing interface defines the encapsulation method used for ATM cells as they are sent onto the wire. The default framing interface is SONET, but the SunATM software also supports the SDH interface. Your switch product information should indicate which interface your switch uses.

Previous versions of the SunATM software allowed you to choose a framing interface for the entire system (by setting a variable in the `/etc/system` file). In the SunATM software, the system variable can be used to allow backwards compatibility, but the preferred method is to select the framing interface for each

interface and to add an entry in the `/etc/opt/SUNWconn/atm/atmconfig` file. An entry in `/etc/opt/SUNWconn/atm/atmconfig` overrides a variable set in `/etc/system` for a particular interface. If there is no value in either `/etc/system` or `/etc/opt/SUNWconn/atm/atmconfig`, the default framing interface is SONET.

Framing entries in `/etc/opt/SUNWconn/atm/atmconfig` should appear on individual lines, with two fields. The first field indicates the interface, `ba`*N*, where *N* is the instance number (for example: `ba0`). The second is either `SDH` or `SONET`, depending on the desired setting.

Example of an `/etc/opt/SUNWconn/atm/atmconfig` File

The following sample `atmconfig` file creates this configuration:

- A LAN Emulation interface `lane0`, supporting UNI 3.1, on the `ba0` interface.
- An interface that supports both Classical IP and LAN Emulation on `ba1`, using UNI 4.0. The LAN Emulation interface name is `lane1`.
- A Classical IP interface, supporting UNI 3.0, on `ba2`, which uses the SDH framing interface.

```
#Interface  UNI/Framing   CIP_Host    LANE_Instance    LANE_Host
#-------------------------------------------------------------
 ba0        3.1          -                0          atm0
 ba1        4.0          atm1             1          atm2
 ba2        3.0          atm3             -          -
 ba2        SDH
```

Configuring a Classical Internet Protocol Interface

Classical Internet Protocol (Classical IP), specified by RFC 1577, is one way of supporting the TCP/IP and UDP/IP protocols in an ATM environment. In Classical IP, an ATM ARP server is used to resolve IP addresses to ATM addresses, replacing the traditional ARP protocol. In this configuration, each host must register with the ARP server when the ATM interface is brought up. For more information on the Classical IP protocols, see “Classical Internet Protocol” on page 47.

ATM ARP is used instead of the traditional ARP because ATM does not support broadcast (a network capability providing transmission from one point to all points on a network). Because Classical IP over ATM does not support broadcast, you cannot use the `ypbind` UNIX command with the `-broadcast` option to automatically locate the NIS server (ypserver) on a Classical IP ATM subnet.

If you are planning to run NIS over your ATM network, use the `ypinit -c` command to specify the list of NIS servers (ypservers). See the `ypinit(1M)` man page for details of setting up the ypserver. Be sure that the IP addresses of the ypservers are listed in the `/etc/hosts` file.

The Routing Information Protocol (RIP) also uses the broadcast feature of IP, so it is not supported under the Classical IP environment. In the Solaris operating environment, RIP is implemented by the daemon `in.routed`.

If you are using Classical IP only, you must explicitly add the routes to the routers in the ATM subnet. You can also specify one router as the default router to provide connectivity outside of the ATM subnet. See the `route(1M)` man page for information on using the `route` command to add specific router entries and to add a default router.

Editing the `/etc/opt/SUNWconn/atm/aarconfig` File

The `/etc/opt/SUNWconn/atm/aarconfig` file is a generic file that must appear on every SunATM system which is supporting Classical IP interfaces. It allows you to specify IP to ATM address translation, permanent virtual circuits (PVCs) to destinations, and the address of the ATM ARP server. The environment allows for a mix of PVCs and switched virtual circuits (SVCs).

Each time the `/etc/opt/SUNWconn/atm/aarconfig` file is modified, run the ATM ARP setup program, `aarsetup`, which is in the `/etc/opt/SUNWconn/bin` directory.

Every node, or client, has both an IP address and either an ATM address or a virtual circuit identifier (VCI). See “ATM Address Resolution” on page 47, for ATM addressing scheme information.

In the IP-ATM address table shown in the `/etc/opt/SUNWconn/atm/aarconfig` file:

- `Interface` is the last part of the device name (`ba0`, for example).
- `Hostname` is either an IP address in “dot” notation or the name of a host that should be locally available unless a non-ATM network connection also exists.
- `ATM Address` consists of 20 octets with each octet represented by a one- or two-digit hexadecimal number and separated by colons.

- The `VCI` field is a positive decimal integer.
- An unused field is denoted by a hyphen.

TABLE 3-2 lists the flags, including configuration flags, and the options they provide.

TABLE 3-2 `/etc/opt/SUNWconn/atm/aarconfig` File Flag Descriptions

Flag	Description
`l`	Represents the ATM address of the local interface on ARP clients or systems not using an ARP server for ATM address resolution, and can be used to assign an ATM address to the host. *Hostname* should not appear; *ATM Address* should be provided if, and only if, SVCs are used. If you provide an *s* entry to use an ARP server (see below), you must also provide an *ATM Address* (a server is meaningful only in an SVC environment). See TABLE 3-3.
`L`	Represents the ATM address of the local interface on an ARP server. *Hostname* should not appear; *ATM Address* is required. See TABLE 3-3.
`s`	Specifies a connection to the ATM ARP server. Either *ATM Address* or *VCI* (in the case of a PVC connection) should appear, but not both. *Hostname* should not appear. The *s* entry is required on all clients that need to communicate with the server for ATM address resolution. See TABLE 3-3.
`t`	Represents an IP to ATM address / VCI entry. `aarsetup` adds these entries into the local table. Any *t* entries on the server must contain *ATM Address* and may also contain *VCI* if PVC communication between the server and client is desired. In addition, there are some cases when a *t* entry may be useful on an ARP client system. If a client wants to communicate with another system over PVCs, the PVC to be used is provided in a *t* entry containing *VCI*; or if a client wishes to cache frequently used addresses to avoid frequent ARP requests, a *t* entry containing *ATM Address* may be provided. See TABLE 3-3.

TABLE 3-2 `/etc/opt/SUNWconn/atm/aarconfig` File Flag Descriptions *(Continued)*

Flag	Description
t *cont.*	Note: If your naming service (NIS+ or DNS) server is an ATM host, you must provide the hostname to IP address resolution for the hosts included in *t* entries, either by using the IP address in the *Hostname* field of the *t* entry, or by adding an entry to the local `/etc/hosts` file.
a	Represents an address that may have access to this host. If no *a* entries appear in the `aarconfig` file, access to the host is unrestricted. Including *a* entries allows access to be restricted to known hosts. As an alternative to listing individual addresses, the ATM address field may contain a prefix, followed by the wildcard `$anymacsel`, which matches any 7-byte ESI/Selector combination following the given prefix. This allows access by any host connected to the switch that is specified by the given prefix. *Hostname* and *VCI* should not appear; *ATM Address* is required. See TABLE 3-3.
m	Notifies the system that the entire ATM address, including the network prefix, must be configured manually on this interface. If your interface is connected to a switch that does not support ILMI address registration, you must include this option in your `/etc/opt/SUNWconn/atm/aarconfig` file. Note that you may not use the variables `$myaddress`, `$prefix`, and `$localswitch_server` (which use the switch prefix obtained from the switch via ILMI) if ILMI address registration is disabled.

Note – Although SunATM supports PVC connections to a server for ARP traffic, RFC 1577 does not specify this case. For interoperability with other implementations, connections to the server should use SVCs.

Note – For two hosts to communicate over PVCs, corresponding PVC connections must also be established in the ATM switch fabric.

TABLE 3-3 describes the required, optional, and illegal fields for each flag type. If a field is unused, it is represented by a hyphen.

TABLE 3-3 /etc/opt/SUNWconn/atm/aarconfig File Flag Requirements and Options

Interface *	Host	ATM Address	VCI	Flags	*
required	illegal	SVC only	illegal	l	local information
required	illegal	required	illegal	L	local information on server
required	illegal	required	illegal	a	access list entry
required	required	or*	or*	t	permanent table entry
required	illegal	xor**	xor**	s	server address/PVC
required	illegal	illegal	illegal	m	manual address registration

*or – Means one or the other required, but using both is also legal.
**xor – Means one or the other required, but using both is illegal.

Note – Group entries in the aarconfig file in a designated order: the local (*l* or *L*) entry first, followed by any other flags in any order. You only need to maintain the ordering within entries for each physical interface; for example, all of the ba0 entries can appear first, and then all of the ba1 entries, etc.

Using Variables in the /etc/opt/SUNWconn/atm/aarconfig File

Because the prefix portion of an ATM address specifies the ATM switch, a number of hosts specified in an aarconfig file can have ATM addresses that share the same prefix. To simplify setting up the aarconfig file, you can define variables that contain part of an ATM address.

A variable's name is an identifier consisting of a collection of no more than 32 letters, digits, and underscores. The value associated with the variable is denoted by a dollar sign followed immediately by the variable name.

Note – Use variables in the ATM address field only. They are not valid in any of the other fields in an entry.

You can use a colon to concatenate multiple variables to represent a single ATM address expression. Thus, if one variable, *v1*, is set to `11:22` and another variable, *v2*, is set to `33:44`, the sequence `$v1:$v2` represents `11:22:33:44`. You can include hexadecimal numbers with variables in the expression. The expression `45:$v1:$v2` would have the value `45:11:22:33:44`.

Use the following format to define variables in the `aarconfig` file:

```
set VARIABLE = EXPRESSION
```

where *VARIABLE* is the name of a variable and *EXPRESSION* is an expression concatenating one or two-digit hexadecimal numbers or the values of variables that have been previously defined. The equal sign is optional, but separate the variable and expression either by white space (spaces or tabs), an equal sign, or both.

Several predefined variables are built into the SunATM software. These variables are summarized in TABLE 3-4.

Note – You can not use the `$prefix` variable or any other variables that use it (including `$myaddress` and `$localswitch_server`) on interfaces that are not running ILMI.

TABLE 3-4 Predefined SunATM Variables

Variable	Description
`prefix`	The 13-byte prefix associated with the local switch.
`mac`	The 6-byte medium access control (MAC) address associated with the local host or interface.
`sel`	The default 1-byte selector for the local interface.
`macsel`	The concatenation of `$mac:$sel`.
`myaddress`	The concatenation of `$prefix:$mac:$sel`, resulting in the default address for the local interface.
`anymac`	A wild card representing any 6-byte ESI. Should only be used in *a* entries.
`anymacsel`	A wild card representing any 7-byte ESI and Selector combination. Should only be used in *a* entries.

TABLE 3-4 Predefined SunATM Variables *(Continued)*

Variable	Description
?	A wild card matching one or two hexadecimal digits within any colon-separated field. For example, `$prefix:$anymac:?` is equivalent to both `$prefix:$anymac:??` and `$prefix:$anymacsel`. However, it is *not* the same as `$prefix:$anymacsel:0?`, which requires that the first digit of the selector byte is a `0`. This wild card should only be used in *a* entries.
`sunmacsel`*N*	The concatenation of one of a series of reserved MAC addresses and `$sel` to create a block of reserved ATM ARP server addresses. *N* should be a decimal number in the range 0-199.
`localswitch_server`	The concatenation of `$prefix`, a unique reserved MAC address, and `$sel`. When used as a server address, restricts server access to clients connected to the local switch only.

In most network configurations, the ATM address assigned to the local interface is `$myaddress`; using this variable in the *l* entry makes it possible to use identical `aarconfig` files on all Classical IP clients using a given server.

The `sunmacsel`*N* variables can be used in conjunction with a prefix, as well as with known server addresses that are not bound to a particular system. As an example, consider the case where a server that supports 50 clients fails. If the ATM address of the server is specific to that particular server, you must change the *s* entry on all 50 clients to switch to a backup server. However, if the ATM address used for that server is `$prefix:$sunmacsel3`, this address is not only guaranteed to be unique, since it uses reserved medium access control (MAC) addresses, you can also simply assign that address to the backup server on the same switch by changing the *l* entry to an *s* entry on one system and bringing up a new server with no changes to the clients.

Note – The `sunmacsel`*N* variables do not include a prefix since a client and server may be on different switches and thus have different local prefix values.

In the case of a single-switch network, you can use `localswitch_server` as a well-known server address that includes the prefix associated with the local switch and a MAC address. It will restrict server access to clients on the local switch and provide a unique ATM address among all ATM clients connected to that switch. Thus, any host with a network prefix other than that of the local switch will be refused a connection to the ARP server if the ARP server's address is `$localswitch_server`.

Several rules apply to the use of variables in the `aarconfig` file:

- Two variables cannot follow each other in an expression without an intervening colon. Thus `$v1:$v2` is legal while `$v1$v2` is not.

- Fields in each line in the `aarconfig` file are separated by white space. Therefore, variables should not be separated from the rest of an ATM address with white space. For example, `$v1: $v2` is illegal.
- Once a variable is defined by a set command, it may not be redefined later in the same configuration file.
- The reserved variable names cannot be set. These names include `prefix`, `mac`, `sel`, `macsel`, `myaddress`, `anymac`, `anymacsel`, `sunmacsel`*N* (where *N* is a number between 0 and 199), and `localswitch_server`.

Note – The ESI portion of `localswitch_server` and the `sunmacsel`*N* variables is a reserved MAC address. The hexadecimal values of the reserved addresses are:

```
localswitch_server   08:00:20:75:48:08
sunmacsel N base     08:00:20:75:48:10
```

To calculate the ESI portion for a `sunmacsel`*N* address, simply add the value of *N* (converted to a hexadecimal number) to the `sunmacsel`*N* base address. For example, the ESI portion of `sunmacsel20` would be 08:00:20:75:48:10 + 0x14 = 08:00:20:75:48:24.

Sample Classical IP Configurations

The following examples demonstrate entries in the `/etc/opt/SUNWconn/atm/aarconfig` file for several typical network configurations.

Although some of the examples show only one sample `aarconfig` file, similarly configured files must appear on each system. Example 2 shows the files for each of the three systems in the configuration.

1. SVC-only: Clients use the default address and access to the ARP server is restricted to clients on the local switch only.

   a. The `/etc/opt/SUNWconn/atm/aarconfig` file on a client:

```
Interface   Host  ATM Address             VCI Flag
ba0          -    $myaddress              -   l
ba0          -    $localswitch_server     -   s
```

   b. The `/etc/opt/SUNWconn/atm/aarconfig` file on the server:

```
Interface Host ATM Address          VCI  Flag
ba0       -    $localswitch_server   -      L
```

2. PVC-only: *hosta* is connected to *hostb* and *hostc* over PVCs. There is no ARP server.

   a. `/etc/opt/SUNWconn/atm/aarconfig` on *hosta*:

```
Interface   Host       ATM Address   VCI  Flag
ba0         -          -               -     l
ba0         hostb      -             100     t
ba0         hostc      -             101     t
```

   b. on *hostb*:

```
Interface     Host    ATM Address        VCI    Flag
ba0           -                 -          -       l
ba0           hosta             -        100       t
ba0           hostc             -        102       t
```

   c. on *hostc*:

```
Interface  Host   ATM Address    VCI   Flag
ba0        -      -                -      l
ba0        hosta  -              101      t
ba0        hostb  -              102      t
```

3. SVC with no ARP server: *hosta* uses SVCs to connect to *hostb* and *hostc*. All hosts are connected to the same switch; there is no ARP server.

```
Interface   Host   ATM Address                     VCI   Flag
ba0         -      $myaddress                       -     l
ba0         hostb  $prefix:08:00:20:d5:08:a8:00    -     t
ba0         hostc  $prefix:08:00:20:21:20:c3:00     -     t
```

4. PVC/SVC mix: *hosta* uses an SVC to connect to *hostb*, and a PVC to connect to *hostc*. *hostb* is not on the local switch; there is no ARP server.

```
Interface  Host  ATM Address
 VCI  Flag
ba0        -     $myaddress                                                -     l
ba0
  hostb 45:00:00:00:00:00:00:00:0f:00:00:00:00:08:00:20:d5:08:a8:00  -     t
ba0        hostc  -
    100   t
```

5. ARP server with Access Restrictions: Hosts are connected to an ATM ARP server that resolves addresses. Access is restricted to the local switch subnet and one additional switch subnet.

   a. `/etc/opt/SUNWconn/atm/aarconfig` on *hosta*:

```
Interface     Host     ATM Address          VCI           Flag
ba0           -        $myaddress             -             l
ba0           -        $prefix:$sunmacsel0    -             s
```

   b. `/etc/opt/SUNWconn/atm/aarconfig` on server:

```
Interface  Host  ATM Address                                          VCI    Flag
ba0         -    $prefix:$sunmacsel0                                  -       L
ba0         -    $prefix:$anymacsel                                   -       a
ba0         -    45:00:00:00:00:00:00:00:0f:00:00:00:00:$anymacsel    -       a
```

6. Manual address configuration: Hosts are connected to a switch that does not support ILMI.

   a. `/etc/opt/SUNWconn/atm/aarconfig` on server:

```
set prfx = 45:00:00:00:00:00:00:00:0f:00:00:00:00
Interface  Host  ATM Address        VCI        Flag
ba0        -     $prfx:$sunmacsel0    -          L
ba0        -     -                    -          m
```

   b. `/etc/opt/SUNWconn/atm/aarconfig` on client:

```
set prfx = 45:00:00:00:00:00:00:00:0f:00:00:00:00
Interface  Host  ATM Address        VCI         Flag
ba0        -     $prfx:$macsel        -           l
ba0        -     $prfx:$sunmacsel0    -           s
ba0        -      -                   -           m
```

Configuring a LAN Emulation Interface

LAN Emulation, standardized by the ATM Forum's LAN Emulation 1.0 specification, is another way of providing TCP/IP and UDP/IP support over an ATM interface. Address resolution information is provided by a series of LAN Emulation services. When a LAN Emulation interface is brought up, it must "join the LAN," that is, it must register with these services. This process, and the address resolution process is described in "LAN Emulation" on page 49.

Unlike Classical IP, the LAN Emulation protocol provides a broadcast service to the upper layer protocols. Therefore, the multicast and broadcast limitations described in "Configuring a Classical Internet Protocol Interface" on page 25, do not affect LAN Emulation interfaces.

Editing the `/etc/opt/SUNWconn/atm/laneconfig` File

The `/etc/opt/SUNWconn/atm/laneconfig` file contains the required configuration information for each interface that uses LAN Emulation. One entry is required for each SunATM interface.

Each time you modify the `/etc/opt/SUNWconn/atm/laneconfig` file, run the LAN Emulation setup program (`lanesetup`). `lanesetup` is in the `/etc/opt/SUNWconn/atm/bin` directory.

Each `/etc/opt/SUNWconn/atm/laneconfig` entry follows this format:

```
Interface         MAC Address/      ATM Address  VCI  Flag
                  ELAN Name
```

These entry fields are described in TABLE 3-5.

TABLE 3-5 `/etc/opt/SUNWconn/atm/laneconfig` Entry Descriptions

Field	Description
Interface	Refers to the LAN Emulation interface, lane*N*.
`MAC Address/ELAN Name`	This field is for the 6-byte MAC address of the interface, or, in the case of an *n* entry, the name of the emulated LAN to join.
`ATM Address`	This field is for the 20-byte ATM address. The `$myaddress` variable assigns the local switch prefix, local MAC address, and default selector.
`VCI`	The VCI field is a positive decimal integer identifying a Permanent Virtual Circuit. Place a dash in this field if VCI is not used.
`Flag`	This field identifies the type of entry. For a complete description of the `laneconfig` flags, see TABLE 3-6.

TABLE 3-6 describes the flags used in the `/etc/opt/SUNWconn/atm/laneconfig` file.

TABLE 3-6 `/etc/opt/SUNWconn/atm/laneconfig` Flag Descriptions

Flag	Description
`l`	This flag designates a local address entry. There must be an `l` entry for each interface running LAN Emulation. The interface and the ATM address must be included; the MAC address may be included (default is the MAC address assigned to the board).
`t`	This flag designates a table entry for the local MAC-ATM address resolution table. If you wish to avoid the address resolution process for a frequently accessed system, for instance, you may include a `t` entry for that system; you may also create PVCs with a `t` entry. The interface, MAC address, and either ATM address or VCI (for SVC or PVC connection, respectively) must be included.
`n`	This flag allows you to specify the name of an emulated LAN to join. By default, the SunATM implementation will use the name provided by the LECS. If you wish to specify a different name, or if your LECS requires that a user include a name in its requests, a name may be provided with this flag. Interface is required; the name should be entered in the second field.
`M`	Notifies the system that a larger MTU size will be used in the ELAN which this lane instance will join. The default MTU size is 1500 bytes. With the M flag, you can set the MTU size to be either 4 Kbytes (4528 bytes) or 9 Kbytes (9218 bytes).

TABLE 3-6 `/etc/opt/SUNWconn/atm/laneconfig` Flag Descriptions *(Continued)*

Flag	Description
`a`	Represents an address that may have access to this host. If no `a` entries appear in the `laneconfig` file, access to the host is unrestricted. Including *a* entries allows access to be restricted to known hosts. As an alternative to listing individual addresses, the ATM address field may contain a prefix, followed by the wildcard `$anymacsel`, which matches any 7-byte ESI/Selector combination following the given prefix. This allows access by any host connected to the switch specified by the given prefix. *Mac Address* and *VCI* should not appear; *ATM Address* is required. See TABLE 3-7.
`c`	This flag allows an alternate LECS address to be specified. By default, the SunATM software uses the well-known address specified in the LAN Emulation standard. If, however, your LECS has a different address, or you wish to connect to the LECS over a PVC, you may provide the alternate ATM address or VCI in a c entry. If you wish to make a PVC connection, the VCI must be 17, as required by the LAN Emulation standard. The interface and ATM address or VCI must be included.
`s`	This flag specifies the LES address or VCI, and instructs the system to contact the LES directly, and to use default subnet configuration information. This flag should be used if your subnet does not have an LECS. Without this entry, the system first connects to the LECS, which provides the LES address and configuration information.
`m`	Notifies the system that the entire ATM address, including the network prefix, must be configured manually on this interface. If your interface is connected to a switch that does not support ILMI address registration, you must include this option in your `/etc/opt/SUNWconn/atm/aarconfig` file. Note that the variables `$myaddress`, `$prefix` and `$localswitch_server` (which use the switch prefix obtained from the switch via ILMI) may not be used if ILMI address registration is disabled.

TABLE 3-7 describes the required, optional, and illegal fields for each flag type.

TABLE 3-7 `laneconfig` Flag Requirements and Options

Interface	MAC Address/ELAN Name	ATM Address	VCI	Flag
required	optional	required	illegal	`l`
required	required	xor[1]	xor[1]	`t`
required	Emulated LAN name	illegal	illegal	`n`
required	MTU Size in bytes	illegal	illegal	`M`
required	illegal	required	illegal	`a`
required	illegal	xor[1]	xor	`c`

TABLE 3-7 `laneconfig` Flag Requirements and Options *(Continued)*

Interface	MAC Address/ELAN Name	ATM Address	VCI	Flag
required	illegal	xor[1]	xor	`s`
required	illegal	illegal	illegal	`m`

[1]xor means that you can use either the ATM Address field or the VCI field, but not both.

Note – Designate unused fields in the `/etc/opt/SUNWconn/atm/laneconfig` file with a dash.

Using Variables in the `/etc/opt/SUNWconn/atm/laneconfig` File

You can use some of the predefined variables from `/etc/opt/SUNWconn/atm/aarconfig` file in the `/etc/opt/SUNWconn/atm/laneconfig`. These variables are listed in TABLE 3-8. For a complete description of how to use these variables, see "Using Variables in the `/etc/opt/SUNWconn/atm/aarconfig` File" on page 29.

Note – Using predefined or user-defined variables in the MAC address field of local ('`l`') entries is not supported. Variables may be used in the MAC address field of other entry types, such as in table ('`t`') entries.

Note – You can not use the `$prefix` variable or any other variables that use it (including `$myaddress`), on interfaces that are not running ILMI.

TABLE 3-8 Predefined SunATM Variables

Variable	Description
`prefix`	The 13-byte prefix associated with the local switch.
`mac`	The 6-byte MAC address associated with the local host or interface.
`sel`	The default 1-byte selector for the local interface.
`macsel`	The concatenation of `$mac:$sel`.
`myaddress`	The concatenation of `$prefix:$mac:$sel`, resulting in the default address for the local interface.

TABLE 3-8 Predefined SunATM Variables *(Continued)*

Variable	Description
anymac	A wild card representing any 6-byte ESI. Should only be used in *a* entries.
anymacsel	A wild card representing any 7-byte ESI and Selector combination. Should only be used in a entries.
?	A wild card matching one or two hexadecimal digits within any colon-separated field. For example, $prefix:$anymac:? is equivalent to both $prefix:$anymac:?? and $prefix:$anymacsel. However, it is *not* the same as $prefix:$anymacsel:0?, which requires that the first digit of the selector byte is a 0. This wild card should only be used in a entries.

Sample LAN Emulation Configurations

The following examples demonstrate entries in the /etc/opt/SUNWconn/atm/laneconfig file for several common configurations.

Although some of the examples show only one sample laneconfig file, similarly configured files must appear on each LAN Emulation client.

1. Basic LAN Emulation client. The ATM and MAC address of a frequently used server is provided. The LECS provides the name of the Emulated LAN.

```
set srvr_mac = 08:00:20:01:02:03

Interface     MAC_Address/     ATM_Address          VCI   Flag
              ELAN Name
lane0            -             $myaddress            -      l
lane0         $srvr_mac        $prefix:$srvr_mac:00 -      t
```

2. LAN Emulation client. The LECS requires that the client send the Emulated LAN name in its messages.

```
Interface   MAC_Address/    ATM_Address    VCI   Flag
            ELAN Name
lane0          -            $myaddress      -     l
lane0       elan1               -            -     n
```

CHAPTER 4

Plumbing and Unplumbing SunATM Interfaces

This chapter describes how to start the SunATM software on your system or use the `atmifconfig` utility to connect and disconnect individual SunATM interfaces without rebooting the system.

Note – You only need to start the SunATM software on your system if you have just installed the SunATM software and one or more SunATM adapters on your system. If the software is already operating on your system, or if you have rebooted your system you do not need to start the software manually.

Starting the SunATM Software for the First Time

This section will describe how to use the drvconfig command and the S00sunatm run control script to start up the SunATM software and load the driver module to the system. When you reboot the system, these commands will be run and start the software correctly.

Plumbing and Unplumbing Individual ATM Interfaces

The `atmifconfig` utility allows interfaces to be added, modified, or removed without rebooting the system by setting up and tearing down the streams necessary to use an ATM device. This setting up and tearing down of streams is referred to as *plumbing* and *unplumbing*.

The two required parameters for the utility are an interface name and plumbing instructions. If an interface is being set up for native ATM applications (no TCP/IP) or for Classical IP, the interface name should have the format `ba`*N*, where *N* is the instance number. For LAN Emulation instances, the interface should have the format `lane`*N*, where *N* is the LAN Emulation instance number.

The second required parameter is the plumbing instruction. There are four possible values: plumb, unplumb, plumb-all, and unplumb-all. The plumb and unplumb options will set up or tear down the specified interface. The plumb-all and unplumb-all options will set up or tear down all ATM instances on the specified physical interface. In this case, the interface parameter must be a `ba`*N* value, since this specifies the physical interface name. This will set up or tear down all the ATM instances on this physical interface, including up to 16 LAN Emulation instances and the Classical IP instance.

A summary of the parameter options for the `atmifconfig` utility is provided in TABLE 4-1.

TABLE 4-1 Parameter Options for `atmifconfig`

Utility	Interface Name	Plumb Instruction
`atmifconfig`	baN	plumb
`atmifconfig`	baN	unplumb
`atmifconfig`	laneN	plumb
`atmifconfig`	laneN	unplumb
`atmifconfig`	baN	plumb-all
`atmifconfig`	baN	unplumb-all

The following example shows the use of `atmifconfig` and the output of `ifconfig -a` before and after the utility is run.

```
# ifconfig -a
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
        inet 127.0.0.1 netmask ff000000
ba0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 9180
        inet 129.144.234.12 netmask ffffff00 broadcast 129.144.234.255
        ether 8:0:20:84:e5:31
lane0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
        inet 129.144.161.12 netmask ffffff00 broadcast 129.144.161.255
        ether 8:0:20:84:e5:31
# atmifconfig ba0 unplumb
# ifconfig -a
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
        inet 127.0.0.1 netmask ff000000
lane0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
        inet 129.144.161.12 netmask ffffff00 broadcast 129.144.161.255
        ether 8:0:20:84:e5:31
# atmifconfig lane0 unplumb
NOTICE: lane0: leaving ELAN
# ifconfig -a
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
        inet 127.0.0.1 netmask ff000000
# atmifconfig ba0 plumb-all
q93b on ba0: Data Link Up
Configuring ATM interfaces: ba0
Configuring ATM LAN Emulation interfaces: lane0
```

CHAPTER 5

Classical IP and LAN Emulation Protocols

ATM is a connection-oriented network protocol, which means that a connection must be established between two communicating entities before data transfer can begin. IP is inherently connectionless. The implementation on the host must therefore reconcile the differences in these two paradigms.

There are two standard ways of doing this: Classical IP, standardized in RFC 1577, and LAN Emulation, standardized in the LAN Emulation 1.0 specification from the ATM Forum. The SunATM architecture supports both of these methods. This chapter discusses some of the key ideas of these two methods.

Both methods allow IP to run transparently over the ATM interface. Thus IP itself sees the ATM interface just as it sees any traditional network interface. Every SunATM interface has a subnet IP address. As an ATM interface starts up, appropriate modules and drivers are plumbed. All the TCP/IP and UDP/IP applications run without modifications over these modules, and all the utilities associated with the network interfaces also run without modification and display similar results (for example, `netstat` and `ifconfig` utilities), with one exception. Because of the different plumbing of the ATM modules, the plumb and unplumb options of `ifconfig` will not work on ATM interfaces. The `atmifconfig(1M)` command may be used to plumb and unplumb ATM interfaces. IP treats the ATM interface as a subnet, choosing the interface used to send a packet out based on the IP address of the destination and on the IP address and netmask of the interface itself.

The transparency to IP is enabled in different ways in Classical IP and LAN Emulation. Those differences will be discussed in later sections of this chapter.

SunATM signalling conforms to the user network interface (UNI) specification of the ATM Forum. Versions 3.0, 3.1, and 4.0 of that specification are supported. This signalling, called Q.2931, runs on top of QSAAL and uses VC 5 for signalling as specified in the Forum specification.

ATM Addresses and Address Registration

UNI signalling uses ATM addresses for signalling. Every ATM interface has an ATM address in addition to its IP address.

ATM addresses, like Network Service Access Point (NSAP) addresses, are 20 octets long. The End System Identifier (ESI) field within the ATM address is a unique 6 octet value; this can be the IEEE hardware MAC address conventionally associated with every network interface. The Selector field is one octet long. The 13 octets that make up the rest of the ATM address are called the Network Prefix, and are derived from the ATM switch fabric to which the interface is connected. Every ATM switch fabric is configured with a 13 octet prefix.

On a SunATM host, the prefix associated with the local switch fabric is represented by the `$prefix` variable. Its value is obtained by the system at configuration time.

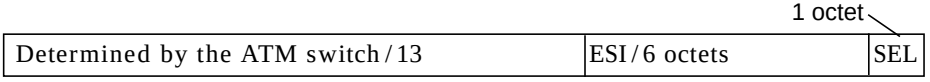


FIGURE 5-1 ATM Address Fields

The UNI specification specifies the Interim Local Management Interface (ILMI) service interface for a client to learn and register its ATM address. The ILMI service interface is based on the use of SNMP over AAL5. In the SunATM software package, ILMI service is provided by an address registration daemon, `ilmid`.

ATM Address Registration Daemon (`ilmid`)

Address registration with a switch is controlled by `ilmid`. When an ATM interface is brought up at boot time, `ilmid` is also started. `ilmid` then begins an exchange of messages with the switch: relaying local address information (the 7 octet ESI and selector) to the switch and receiving the 13 octet network prefix information from the switch.

The default local address that is registered with the switch at boot time consists of the network prefix provided by the switch, the MAC address assigned to the local interface, and the default selector for that interface (usually 0). Additional addresses are registered in two different ways. `aarsetup(1M)` and `lanesetup(1M)` register additional local addresses that may appear in `aarconfig(4)` and `laneconfig(4)`, respectively. You can also use `atmreg(1M)` to register addresses, un-register addresses, and check the status of any address.

Classical Internet Protocol

For ATM to work transparently under IP, an IP address must be resolved to an ATM address and a connection to that destination must be established. Classical IP does this via a database of IP / ATM address pairs that is either provided by an ATM ARP server that is accessible to all hosts on the subnet, or is maintained locally in each host.

ATM Address Resolution

Traditional TCP / IP and UDP / IP applications use IP addresses for communicating to a destination. For these applications to run like traditional applications, IP addresses need to be resolved into ATM addresses. The ATM address then signals to establish an ATM connection to the destination. An ATM connection in turn is represented by a VPI / VCI. The host must use this returned VPI / VCI to send packets to the destination that represents the ATM connection.

ATM address resolution, also called ATM ARP, follows RFC 1577, the classic draft that describes the ATM ARP process.

RFC 1577 assumes the existence of an ATM ARP server on every subnet. Every client on the subnet communicates with the ATM ARP server to derive the destination's ATM address from its IP address. The ATM ARP server holds the IP-to-ATM address information for all hosts in the ATM subnet. It is likely that initial ATM configurations will not rely on dynamic ATM address resolution because it requires the presence of an ATM ARP server on every subnet. Also, there are no specified standards for providing redundant ATM ARP servers for a subnet. As specified, the ATM ARP server would constitute a single point of failure in the system. From a practical standpoint, however, early configurations can use an IP-to-ATM address database in every system, thus avoiding the IP-to-ATM address resolution step altogether.

The RFC requires a router for passing data between subnets. SunATM software provides ATM utilities that allow configurations to specify IP-to-ATM addresses in `/etc/opt/SUNWconn/atm/aarconfig` files. The `aarsetup` program uses the information in `/etc/opt/SUNWconn/atm/aarconfig` to create IP-to-ATM address resolution tables. Dynamic entries into a server's resolution table are also supported.

TABLE 3-1 on page 24 shows the format of the `/etc/opt/SUNWconn/atm/aarconfig` file for specifying the IP-to-ATM address. It is important for the file to be consistent on all systems in the subnet. See "Editing the `/etc/opt/SUNWconn/atm/aarconfig` File" on page 26.

ATM ARP Address Resolution Tables

Depending on the `aarconfig` file, the Classical IP software runs as either a server or a client. As a server, the Classical IP software handles ATM ARP requests originating from its clients. An ATM server has to be configured for each subnet. The ATM ARP server code conforms to RFC 1577: clients send ATM ARP requests to the server to resolve a destination IP address to an ATM address. The server then replies to ATM ARP requests by sending an ATM ARP response. If the server does not have the IP-to-ATM address entry, then it replies with NAK.

All the IP-to-ATM address entries specified in the `/etc/opt/SUNWconn/atm/aarconfig` file are entered into a kernel resident table by the ATM ARP setup program, `aarsetup`. Additional entries in the kernel table are added dynamically using the inverse ARP process. When a client connects to the server, the server sends an inverse ARP request back to the client to obtain the client's IP address. When a response is received, an entry is created for that client. The Classical IP software also responds to client ARP requests. The software looks up a kernel IP-to-ATM address entry and responds to an ATM ARP request with either an ATM ARP reply or ATM ARP NAK (if there is no entry in the table). Note that an ATM ARP client uses the virtual channel (VC) specified in the `/etc/opt/SUNWconn/atm/aarconfig` file to communicate with the server; or, if an ATM address is specified, it establishes a switched virtual circuit (SVC) connection to communicate with the server.

While dynamic entries in the ARP server's table make network administration less complex, they also create a security problem. Any host can register with the ARP server and therefore gain access to the subnet. To resolve this issue, you can provide a list of hosts or networks with `a` entries in the server's `/etc/opt/SUNWconn/atm/aarconfig` file. If no `a` entries appear, any host can connect to the server. If any `a` entries exist, only those hosts whose addresses match those specified will be allowed to connect.

Although the `a` entry requires a complete ATM address, you can reference multiple addresses in a single entry using the provided wildcards. See "Using Variables in the `/etc/opt/SUNWconn/atm/aarconfig` File" on page 29 for more information about this feature.

The advantage of having an ATM ARP server in the subnet is that it represents a known source for all address resolutions. It is the only host that a client must know about to have IP addresses resolved to ATM connections, and it allows for access control in the ATM network.

When the `/etc/opt/SUNWconn/atm/aarconfig` file has been modified on a system, it is necessary to rerun `aarsetup`.

Note – For better caching, all clients have the option of adding to their configuration file the IP-to-ATM address information for other clients. This can benefit clients that communicate frequently because it eliminates having to go through the ATM ARP server for IP-to-ATM address resolution.

If a host has multiple SunATM cards, the host can be a server for one IP subnet and a client for another. This is handled transparently by `aarsetup`.

LAN Emulation

As described in previous sections, Classical IP provides its own (IP-to-ATM) address resolution mechanism that corresponds to and replaces ARP, thus allowing IP-based applications to run transparently over ATM. A shortcoming of Classical IP, and a primary reason it must replace the traditional ARP, is that it does not support broadcast messages.

Because ATM is a connection-oriented protocol (unlike ethernet), implementing broadcast is much more difficult. The only host that receives a message is the host to which the message is addressed, and a call must be established to that host before the message can be sent.

Local Area Network (LAN) Emulation, as standardized by the ATM Forum, provides mechanisms to send broadcast messages in an ATM environment. Given this capability, LAN Emulation is also able to work transparently with ARP, as well as IP. IP and ARP send broadcast messages over the ATM interface, and thus resolve IP addresses to MAC addresses; messages are then sent to the LAN Emulation driver, which has its own address resolution protocol (similar to that of Classical IP) to resolve the medium access control (MAC) address to an ATM address and connection.

The SunATM software implements the client side of the LAN Emulation standard. To use LAN Emulation in an environment, several LAN Emulation services must also exist in the emulated LAN. These services, called the LAN Emulation Configuration Server (LECS), the LAN Emulation Server (LES), and the Broadcast and Unknown Address Server (BUS), are generally provided in an ATM switch. The following sections provide an overview of the functions of these servers.

LAN Emulation Servers

There are three types of LAN Emulation servers. Each type is briefly described in this section.

LAN Emulation Configuration Server

This server is first contacted by a host interface when the host is brought up on the emulated LAN. Its address is generally a well-known address specified by the LAN Emulation standard that is coded into the host software; thus establishing this connection requires no input from you. When contacted by a host wishing to join its emulated LAN, the LECS replies with configuration parameters for the emulated LAN, as well as the address of the LES.

LAN Emulation Server

The second step in joining an emulated LAN is to make a connection to the LAN Emulation Server. After receiving the LES address from the LECS, a host will establish a connection to the LES. The LES may add the host to a point-to-multipoint call that is maintained by the LES with connections to every host in the emulated LAN. This point-to-multipoint connection, if created by the LES, is used to send control information to each host on the emulated LAN.

The LES acts as the ATM ARP server. Since IP and ARP work with MAC addresses, an additional address resolution step is required to convert a MAC address to the corresponding ATM address, which is used to make a connection to the target host; this resolution step is provided by the LES.

Broadcast and Unknown Address Server

The final step in joining an emulated LAN is to make a connection to the BUS. The ATM address of the BUS is obtained by sending a LAN Emulation ARP request to the LES for the broadcast address. Once established, this connection is used to send broadcast messages to the BUS, which will add the client to a point-to-multipoint call including all hosts on the emulated LAN. Thus when a broadcast message (such as an IP ARP request) is received by the LAN Emulation host from its upper layers, it sends that message to the BUS, which forwards it to all hosts in the emulated LAN. Just as in the case of ethernet, the correct host responds to the sender, and thus the IP address is resolved to a MAC address.

Resolving an IP Address to an ATM Connection

The entire process from the time IP sends a message addressed to an IP address to the arrival of that message at the appropriate destination was hinted at in the previous descriptions of the LAN Emulation servers. To demonstrate how those pieces work together during the actual transmission of a message, the process is described below. This description assumes that none of the needed addresses have

been previously resolved and cached. The two hosts involved are referred to as the source (the system that wishes to send a message) and the target (the system to which the message is addressed).

1. IP has a message to transmit and only knows the IP address of the target system. IP first sends a message to ARP, to resolve the IP address to a MAC address.
2. ARP creates a broadcast request for the MAC address corresponding to the given IP address, which it sends to the LAN Emulation driver.
3. The LAN Emulation driver recognizes that this message has a broadcast address, and sends it to the BUS, which forwards the message to every host on the emulated LAN.
4. The message is received on each host, and sent up to ARP by the LAN Emulation driver.
5. On the target, ARP recognizes the IP address as its own and sends a response with its MAC address (addressed to the source's MAC address) down to the LAN Emulation driver.
6. The LAN Emulation driver sends an LE ARP request to the LES to resolve the source's MAC address to its ATM address.
7. The LES responds with the requested ATM address, and the target host sets up an ATM connection to the source host, over which it sends the IP ARP response.
8. The LAN Emulation driver on the source receives the IP ARP response message and sends it up to ARP. ARP then inserts the MAC address into the original message and sends it back down to the LAN Emulation driver.
9. The LAN Emulation driver then must send an LE ARP request to the LES to resolve the MAC address in the message from ARP to an ATM address. When it receives an LE ARP response, it then sees that it has a connection to that address (established by the target to return the IP ARP response) and sends the original IP message to the target over that connection.

LAN Emulation Connections

There are several connections established at all times when a host is a member of an emulated LAN. The following table outlines the various LAN Emulation-related connections that should be on a LAN Emulation client (LEC).

Note – Use the command `qccstat(1M)` to view all existing connections for a given interface.

TABLE 5-1 LAN Emulation Connections

VCC	Endpoints	Comments
Configuration Direct	LEC —> LECS	This connection is not required to remain open after the initial join of the emulated LAN, and thus may time out after a host has joined the LAN.
Control Direct	LEC —> LES	Point-to-point connection over which the host may send LE ARP requests and receive responses from the LES.
Control Distribute	LES —> LEC	Point-to-multipoint connection over which the LES may send administrative information to all hosts. Hosts may not send on this connection.
Multicast Send	LEC —> BUS	Point-to-point connection over which the host may send broadcast messages to the BUS. A limited amount of data is also allowed on this connection.
Multicast Forward	BUS —> LEC	Point-to-multipoint connection over which the BUS sends broadcast messages. Hosts may not send on this connection.

CHAPTER 6

SunATM and Solaris Networking Features

This chapter discusses the SunATM and Solaris Networking features.

ATM and SNMP

Two of the ATM standards supported by the SunATM software (the User Network Interface (UNI) and LAN Emulation (LANE) specifications) include definitions of SNMP-style Management Information Bases (MIBs) relevant to those standards. These MIBs are referred to as the ATM Forum (ATMF) and LAN Emulation (LANE) MIBs, respectively.

The ATM SNMP daemon (`atmsnmpd`) handles requests for information in both MIBs, as well as the system MIBs, from SNMP-based network management systems (such as the SunNet Manager program), and from `ilmid`, when it is required, for SNMP requests coming from the switch.

`atmsnmpd` can be used as a forwarding agent. If you configure it as a forwarding agent, `atmsnmpd` will forward SNMP requests for unknown MIBs to the port specified with the forward option, `-f`. This allows a system to have two SNMP agents respond to requests received over the SNMP port. FIGURE 6-1 illustrates the required configuration. To set up this example configuration, `atmsnmpd` must be started with the parameter `-f 1000` and `other_snmpd` must be started so that it listens on port 1000.

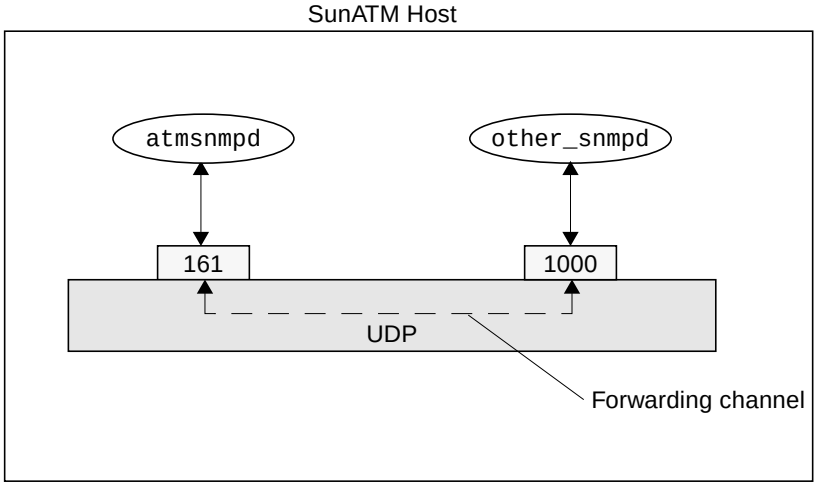


FIGURE 6-1 Using `atmsnmpd` as a Forwarding Agent

Note – If you do not specify a forwarding port for unknown requests, `atmsnmpd` will respond with a "No Such Name" error to requests for MIBs that it does not support. If you do specify a forwarding port, `atmsnmpd` instead forwards the request to that port. Responses received from the agent running on the forwarding port are sent to the requesting SNMP manager with no modification. If the agent does not respond, then `atmsnmpd` does not send a response back.

SNMP and Solaris

The ATM SNMP agent in SunATM supports a framework for SNMP agents; however, this means that its setup depends on the version of the Solaris operating environment in which it is running. This section discusses those differences, which are mostly transparent to the user.

Solaris 2.6, Solaris 7, and Solaris 8 Compatible Software

The Solaris 2.6, 7, and 8 operating environments include a bundled SNMP agent that is designed to run as a *master* agent, binding by default to UDP port 161. Configure other agents to listen to other UDP ports and act as subagents, then configure the master agent to forward particular requests to those subagents. This framework provides a single agent at port 161 with the combined capabilities of the master agent and all the additional subagents.

The SunATM software has been designed to take advantage of this framework if it is installed on a system running Solaris 2.6, 7, or 8. The files necessary for the ATM SNMP agent to be recognized by the master agent (`atm.reg` and `atm.rsrc`) are copied under `/etc/snmp/conf` by the `S00sunatm` startup script if it detects that the system is running Solaris 2.6, 7, or 8. SNMP requests pertaining to the ATM Forum subtree (`atmForum`) are forwarded to the `atmsnmpd` from the master agent. In addition, `atmsnmpd` binds, by default, to port 1000, rather than 161, under Solaris 2.6 or later. The UDP port may still be changed using `atmadmin`, but the default will be 1000 in Solaris 2.6, 7, or 8.

ATM and Logical Interfaces

The SunATM software supports logical interfaces in the LAN Emulation environment. Logical interfaces allow you to assign multiple IP addresses to a single Emulated LAN interface. A logical interface name consists of three parts: the device name (in the case of SunATM LAN Emulation, `lane`); the major number, which corresponds to the lane instance number; and the minor number, which distinguishes the logical interfaces on a single physical interface. The format of a LAN Emulation logical interface name is `lane`*N*`:`*X*, where *N* is the major number and *X* is the minor number.

Each logical interface will be associated with a unique IP hostname and address. All logical interfaces on a given physical interface will be associated with the same ATM and MAC addresses. Configure logical interfaces by placing multiple entries for a given interface in the `/etc/opt/SUNWconn/atm/atmconfig` file.

Consider the following rules when you use logical interfaces with the SunATM software:

- Only one signalling protocol (UNI 3.0, 3.1, or 4.0) is supported per interface, and must appear in the first entry for that interface.
- Only one Classical IP hostname may be assigned to an interface; it can appear in any entry, in any order, in `/etc/opt/SUNWconn/atm/atmconfig`.
- The first `lane`*N* entry on an interface must be for `lane`*N*`:0`, or simply `lane`*N*. `lane`*N* and `lane`*N*`:0` are identical and interchangeable.
- IP limits the number of logical interfaces on a physical interface to 256 (the minor number *X* must be in the range 0 - 255) in Solaris 2.5.1, and to 8194 (the minor number *X* must be in the range0 - 8193) in Solaris 2.6 and later releases.

The following examples show the `atmconfig` and `laneconfig` files and the `ifconfig -a` output for a system with one physical interface, `ba0`. That interface runs both Classical IP and LAN Emulation under UNI 3.1, and has 4 different IP addresses. Configure the hostnames, `cip0`, `atm0`, `atm1`, and `atm2`, appropriately in `/etc/hosts`.

The example `/etc/opt/SUNWconn/atm/atmconfig` file:

```
Interface   UNI  CIP Hostname  LANE Instance   LANE Hostname
ba0         3.1          cip0            0              atm0
ba0          -              -           0:1             atm1
ba0          -              -           0:2             atm2
```

The corresponding example `/etc/opt/SUNWconn/atm/laneconfig` file:

```
Interface   MAC Address/     ATM Address  VCI   Flag
            ELAN Name
lane0           -            $myaddress    -     l
```

The resulting `ifconfig -a` output:

```
# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
ba0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 9180 index 3
        inet 192.29.235.36 netmask ffffff00 broadcast 192.29.235.255
        ether 8:0:20:7a:37:af
lane0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 4
        inet 192.29.240.36 netmask ffffff00 broadcast 192.29.240.255
        ether 8:0:20:8b:6d:d0
lane0:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 4
        inet 192.29.241.36 netmask ffffff00 broadcast 192.29.241.255
lane0:2: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 4
        inet 192.29.242.36 netmask ffffff00 broadcast 192.29.242.255
```

Supporting Multiple Emulated LANs on a Single Interface

The SunATM software allows a single ATM interface to join up to sixteen emulated local area networks (ELANs), provided this is allowed by the switch and LAN Emulation (LANE) services. Each ELAN joined will be represented by a unique lane instance (for example, `lane0` or `lane1`).

Note – A requirement for supporting this feature is that the adapter card be assigned multiple MAC addresses, which is supported in the SunATM/S 2.1 and SunATM/P 3.0 adapters. This feature *does not* work with the older SunATM/S 2.0 adapters. You can find the number of MAC addresses assigned to your SunATM adapter by using the `atmgetmac(1M)` command with the `count` option.

Configure multiple ELANs by placing multiple entries in the `/etc/opt/SUNWconn/atm/atmconfig` and `/etc/opt/SUNWconn/atm/laneconfig` files. Each `lane` instance will have a unique hostname and IP address, ATM address, and MAC address associated with it. In addition, assign an ELAN name to the instance if any ELAN other than the default is to be joined. Provide this information, with the exception of the MAC address, which is retrieved from the board itself, in the `/etc/opt/SUNWconn/atm/atmconfig` and `/etc/opt/SUNWconn/atm/laneconfig` configuration files.

Note – Only one signalling protocol (for example, UNI 3.0 or 3.1) and one Classical IP instance are supported per physical interface. Specify the UNI version in the first `/etc/opt/SUNWconn/atm/atmconfig` entry for a given interface; the Classical IP instance may be specified in any entry.

The following example shows the `/etc/opt/SUNWconn/atm/atmconfig` and `/etc/opt/SUNWconn/atm/laneconfig` files and the `ifconfig -a` output for a system with one SunATM interface, `ba0`. The interface uses UNI 3.0 for signalling, and does not run Classical IP. It joins 4 emulated LANs: the default, `elan1`, `elan2`, and `elan3`.

The example `/etc/opt/SUNWconn/atm/atmconfig` file:

```
Interface    UNI  CIP Hostname   LANE Instance LANE Hostname
ba0          3.0      -               0             atm0
ba0           -       -               1             atm1
ba0           -       -               2             atm2
ba0           -       -               3             atm3
```

The corresponding example `/etc/opt/SUNWconn/atm/laneconfig` file:

```
Interface   MAC Address/    ATM Address   VCI Flag
            ELAN Name
lane0           -            $myaddress     -    l
lane1           -            $myaddress     -    l
lane1         elan1               -         -    n
lane2           -            $myaddress     -    l
lane2         elan2               -         -    n
lane3           -            $myaddress     -    l
lane3         elan3               -         -    n
```

The resulting `ifconfig -a` output:

```
lo0:  flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
      inet 127.0.0.1 netmask ff000000
lane0:flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
      inet 192.29.240.36 netmask ffffff00 broadcast 192.29.240.255
      ether 8:0:20:7a:37:af
lane1:flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
      inet 192.29.241.36 netmask ffffff00 broadcast 192.29.241.255
      ether 8:0:20:7a:37:b0
lane2:flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
      inet 192.29.242.36 netmask ffffff00 broadcast 192.29.242.255
      ether 8:0:20:7a:37:b1
lane3:flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
      inet 192.29.243.36 netmask ffffff00 broadcast 192.29.243.255
      ether 8:0:20:7a:37:b2
```

ATM and IP Multipathing

The SunATM software supports IP Multipathing in the LAN Emulation environment. IP Multipathing allows you to create interface groups which provide stand-by interfaces which IP may failover to in the event of hardware failure between an interface and its network equipment. In addition, it allows increased network throughput by spreading traffic across the group member interfaces

Consider the following rules when you use IP Multipathing with the SunATM software:

- IP Multipathing is only supported on interfaces running LAN Emulation. It cannot be configured on Classical IP interfaces.
- LAN Emulation interfaces that are members of an IP Multipathing group must all be members of the same Emulated LAN. This fulfills the IP Multipathing requirement that the group members be connected to the same IP Link.
- ATM interfaces do not have `/etc/hostname.laneN` files, so this method of configuring IP Multipathing information is not supported on ATM LAN Emulation interfaces. You must instead use the `ifconfig` command, as described in the *IP Network Multipathing Administration Guide*.

APPENDIX **A**

Application Programmers' Interface

The Application Programmers' Interface (API) provided with this software release is an interim API from Sun Microsystems, Inc. that can be used on Sun Platforms.

In the ATM environment, data is sent between hosts over Virtual Circuits (VCs). VCs are point-to-point (or point-to-multipoint) connections between two or more ATM hosts.

VCs can be created in one of two ways:

- Manual configuration at each host and each intermediate network point, also known as Permanent Virtual Circuits (PVC)
- ATM signalling, also known as Switched Virtual Circuits (SVC)

After the VC has been created, the application notifies the SunATM `ba` driver that it is sending and receiving data on the new VC.

- If you are using a PVC, this is the only configuration required on the Sun host.
- If you are using an SVC, there are two required actions:

1. Create the SVC with the Q.93B driver.
2. Establish the data connection with the `ba` driver.

Note – For historical reasons, Q.93B and Q.2931 are used interchangeably.

Using the SunATM API with the Q.93B and the ATM Device Drivers

The architecture illustrated in FIGURE A-1 must be established on a SunATM system in order to perform Q.2931 signalling and send data over established connections. The ATM device driver, SSCOP modules, and Q.93B driver are “plumbed” at boot time. The task remaining for application developers is to create the connections between their application and the Q.93B and ATM device drivers.

Both the Q.93B and ATM device driver are STREAMS drivers; connecting to them is for the most part no different than connecting to other STREAMS drivers. The following sections describe the steps required to connect to each driver, use the drivers to establish ATM connections, and send data over those connections.

For examples of applications that use the SunATM API, see the sample programs installed in `/opt/SUNWconn/atm/examples`.

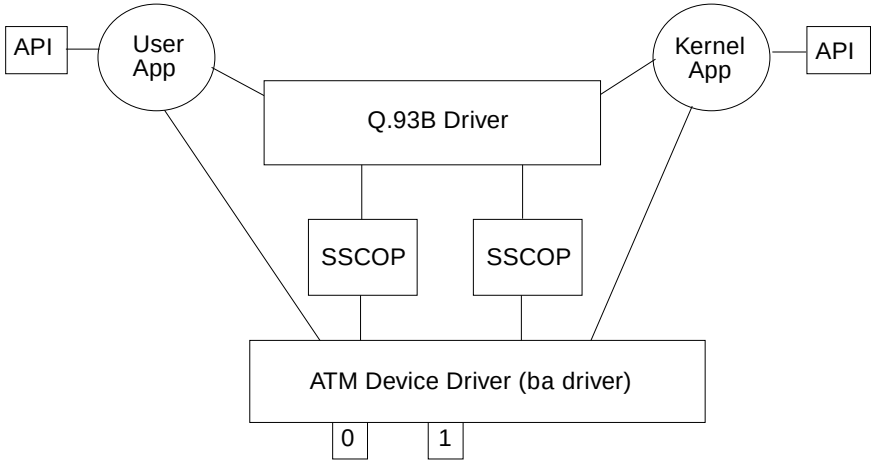


FIGURE A-1 ATM Signalling

Q.93b Driver Interface

The signalling API, called Q.2931 Call Control (`qcc`), consists of two sets of similar functions: one for applications running in the kernel and one for applications running in user space. Each set provides functions to build and parse Q.2931 signalling messages, which are required to set up and tear down connections.

One additional function assists applications in establishing appropriate connections to the Q.93B driver. `q_ioc_bind` associates a service access point (SAP) with the specified connection to the Q.93B driver. The driver uses the SAP to direct incoming messages to applications.

Establishing a Connection to the Q.93B Driver

Use the `open(2)` system call to obtain a file descriptor to the driver. After opening the driver, `q_ioc_bind` should be called, associating in the Q.93B driver a service access point (SAP) with this application. Finally, if the application is a kernel driver, it should be linked above the Q.93B driver, using the `I_LINK` or `I_PLINK` ioctl (refer to the `streamio(7)` man page for information about this `ioctl`).

Setting up an ATM Connection Over a Switched Virtual Circuit (SVC)

After connecting to the Q.93B driver, either by directly calling the functions as a user application or by having a setup program connect your application driver as described in the preceding section, the Q.93B driver is available to your application to establish switched virtual circuits (SVCs) using the Q.2931 signalling protocol. The Q.2931 message set is displayed in TABLE A-1.

TABLE A-1 Messages Between the User and the Q.93B Driver

Message Type	Direction*
SETUP	BOTH
SETUP_ACK	UP
CALL_PROCEEDING	BOTH
ALERTING	BOTH
CONNECT	BOTH
CONNECT_ACK	UP
RELEASE	DOWN
RELEASE_COMPLETE	BOTH
STATUS_ENQUIRY	DOWN

*UP is from Q.93B to user;

DOWN is from user to Q.93B

TABLE A-1 Messages Between the User and the Q.93B Driver *(Continued)*

Message Type	Direction*
STATUS	UP
NOTIFY	BOTH
RESTART	BOTH
RESTART_ACK	BOTH
ADD_PARTY	BOTH
ADD_PARTY_ACK	BOTH
ADD_PARTY_REJECT	BOTH
PARTY_ALERTING	BOTH
DROP_PARTY	BOTH
DROP_PARTY_ACK	BOTH
LEAF_SETUP_FAIL	BOTH
LEAF_SETUP_REQ	BOTH

*UP is from Q.93B to user;

DOWN is from user to Q.93B

The Q.93B driver is an M-to-N mux STREAMS driver. Multiple application programs can be plumbed above the driver, and multiple physical interfaces can be connected below Q.93B. Applications can access any or all of the physical interfaces, and messages received on the physical interfaces can be directed to any of the applications. To direct messages through the Q.93B driver, messages from applications must include a physical interface name to identify the outgoing interface and a SAP to identify the application to which the message should be directed on the receiving host.

Send messages to Q.93B by applications according to the format illustrated in FIGURE A-1; kernel applications use `putnext(9f)` to send the mblocks shown and user applications send two corresponding `strbufs` using `putmsg(2)`.

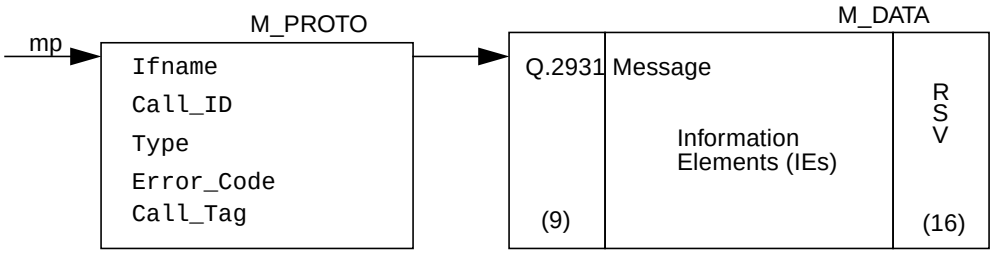


FIGURE A-2 Message Format

TABLE A-2 Fields in the M_PROTO mblock

Message	Explanation
`Ifname`	Null-terminated string containing the device name
`Call_ID`	Unique number from Q.93B for each interface.
`Type`	Same as the Q.2931 message type except there is a local Non-Q.2931 message type `SETUP_ACK`. The `SETUP_ACK` message is used to provide the `Call_ID` to the user.
`Error_Code`	Error returned from Q.93B when an erroneous message is received from the user. The same mblock chain is returned to the user with the `Error_Code` field set. The user must always clear this field
`Call_Tag`	Number assigned by the calling application layer to a `SETUP` message. When a `SETUP_ACK` is received from Q.93B, the `Call_ID` has been set; use the `Call_Tag` field to identify the acknowledgment (ack) with the original request. From that point on, use the `Call_ID` value to identify the call.

The structure included in the M_PROTO mblock is defined as the `qcc_hdr_t` structure in the `<atm/qcctypes.h>` header file. In the second mblock, the Q.2931 header portion (9 bytes) of the Q.2931 message is blank and later filled in by the Q.93B driver. The application should also reserve 16 bytes at the end of the second mblock for the layer 2 (Q.SAAL) protocol performance. The qcc functions can be used to create messages in this format.

The following sections give a brief overview of Q.2931 signalling procedures, from the perspective of an application using the SunATM API. For more details on the procedures, refer to the *ATM Forum's User Network Interface Specification*, version 3.0, 3.1, or 4.0. For further information on the `qcc` functions, which are outlined in TABLE A-3, see the appropriate man pages in Section 3 (for user applications) or Section 9F (for kernel applications). You can find the man pages under the function group name or any specific function name. For example, the man page that documents the `qcc_bld_*` function group may be accessed by one of the following

at a command prompt: **`man qcc_bld`**, **`man qcc_bld_setup`**, or **`man qcc_bld_connect`**. FIGURE A-3 illustrates the message flow during typical call setup and tear down.

TABLE A-3 `qcc` Functions

Name	Functionality	Input	Output
`qcc_bld_*`	Creates and encodes a message; enables customization of a limited set of values, depending on the message type. Configurable values are passed in as parameters.	Parameter values	Encoded Q.2931 message (in the format shown in FIGURE A-2)
`qcc_parse_*`	Extracts a defined set of values from an encoded message	Encoded Q.2931 message (in the format shown in FIGURE A-2)	Parameter values
`qcc_len_*`	Returns the maximum length of the buffer that should be allocated for the second strbuf in a Q.2931 message. Only applicable to user space applications; the kernel API allocates the buffers inside the `qcc_bld/qcc_pack` functions.	none	Maximum length of the message
`qcc_create_*`	Creates a message structure with the required values set. You can further customize the structure using `qcc_set_ie`.	Default parameter values	Message structure (defined in `<atm/qcctypes.h>`)
`qcc_set_ie`	Updates or inserts values for an information element into a message structure.	Message structure and IE structure (defined in `<atm/qcctypes.h>`)	Updated message structure
`qcc_pack_*`	Takes a message structure and encodes it into an actual Q.2931 message, consisting of the two mblks (or strbufs) illustrated in FIGURE A-2.	Message structure (defined in `<atm/qcctypes.h>`)	Encoded Q.2931 message (in the format shown in FIGURE A-2)
`qcc_unpack_*`	The reverse of `qcc_pack_*`: takes an encoded message and decodes the data into a message structure.	Encoded Q.2931 message (in the format shown in FIGURE A-2)	Message structure (defined in `<atm/qcctypes.h>`)

Call Setup

To make a call, send a SETUP message down to Q.93B and wait for a SETUP_ACK from Q.93B. The SETUP message should include a Broadband Higher Layer Information (BHLI) information element that contains a four-octet SAP identified as User Specific Information. The SAP is used to identify the application on the

receiving host to which the Q.93B should direct the message. After receiving a SETUP_ACK with a 0 error field, wait for either a CALL_PROCEEDING, ALERTING, CONNECT, or RELEASE_COMPLETE message from Q.93B (all other messages are ignored by Q.93B). After you receive the CONNECT message, you can use the virtual channel.

Respond to a SETUP message from Q.93B with either a CALL_PROCEEDING, ALERTING, CONNECT, or RELEASE_COMPLETE message to Q.93B. After yu receive the CONNECT_ACK message, you can use the virtual channel.

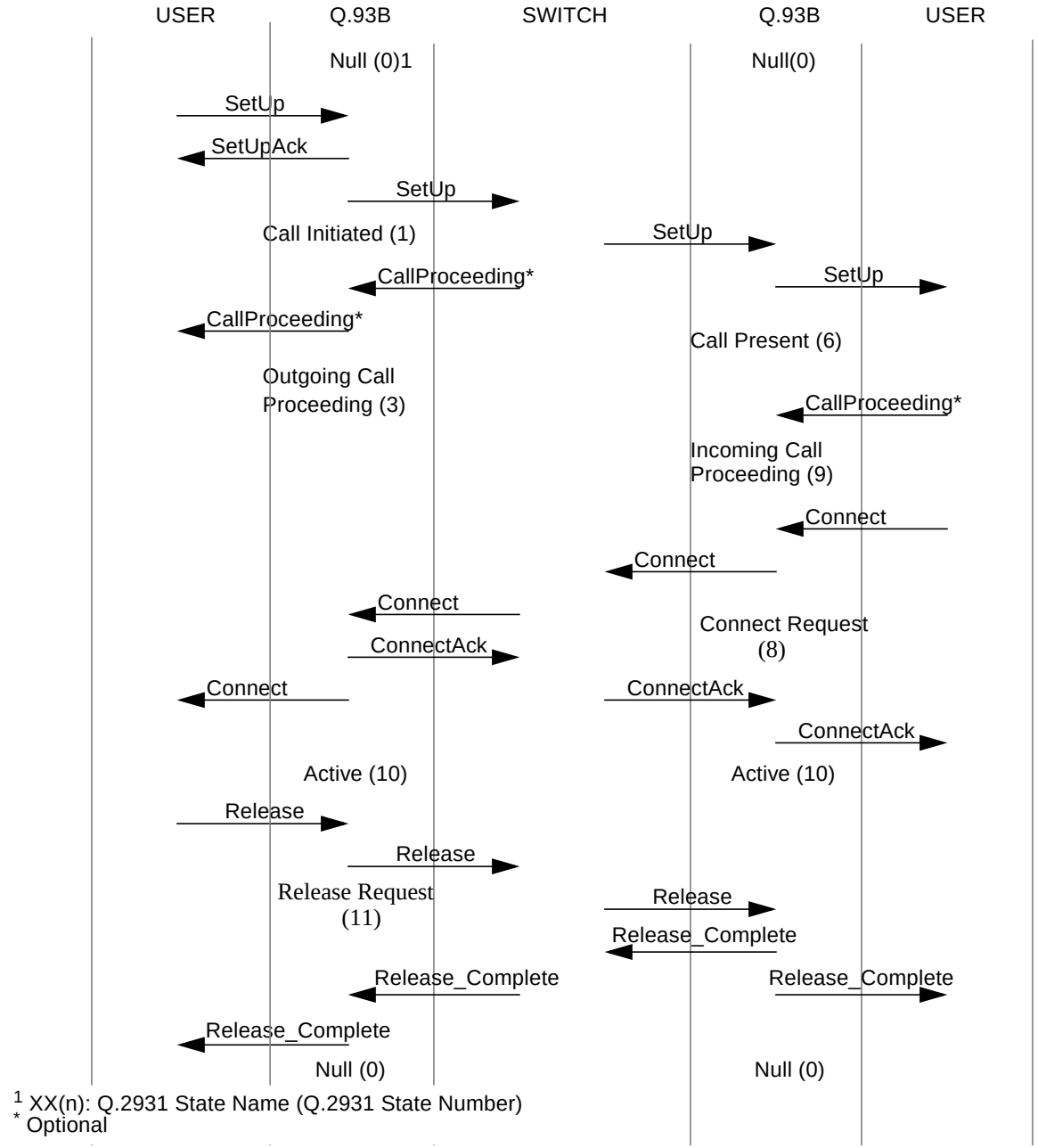
Release Procedure

To clear an active call or a call in progress, send a RELEASE message down to Q.93B and wait for a RELEASE_COMPLETE from Q.93B. Any time you receive a RELEASE_COMPLETE message from Q.93B, release the virtual channel if the call is active or in progress.

Q.93B never sends a RELEASE message to the end user; it will always send a RELEASE_COMPLETE. Only send the RELEASE_COMPLETE message when rejecting a call in response to a SETUP message from Q.93B. At any other time, to reject or tear down a call, send a RELEASE message to Q.93B.

Exception Conditions

If for any reason Q.93B cannot process a SETUP message received from an end user, the SETUP_ACK is returned with an error value set, and call setup is not continued. The error value will be one of the cause codes specified in the ATM Forum UNI standard.



[1] XX(n): Q.2931 State Name (Q.2931 State Number)
* Optional

FIGURE A-3 Message Flow for Normal Call Setup and Tear Down

Connecting, Sending, and Receiving Data with the ATM Device Driver

Connecting to the ATM device driver involves several steps, some of which include several `ioctl` calls. To create a more standardized interface for user space applications, a set of `atm_util` functions is available to application writers. An overview of those functions is provided in TABLE A-4. For more detailed information, refer to the `atm_util(3)` man page. The `ba(7)` man page contains a more detailed discussion of the driver-supported `ioctl`s.

TABLE A-4 `atm_util` Function Overview

Name	Functionality	Kernel Equivalent
`atm_open`	Opens a stream to the ATM device driver	Must be done by a user space setup program
`atm_close`	Closes a stream to the ATM device driver	Must be done by a user space setup program
`atm_attach`	Attaches to a physical interface	Must be done by a user space setup program
`atm_detach`	Detaches from a physical interface	Must be done by a user space setup program
`atm_bind`	Binds to a Service Access Point	send `DL_BIND_REQ`
`atm_unbind`	Unbinds from a Service Access Point	send `DL_UNBIND_REQ`
`atm_setraw`	Sets the encapsulation mode to raw	Send `DLIOCRAW`
`atm_add_vpci`	Associates a vpci with this connection	`A_ADDVC ioctl`
`atm_delete_vpci`	Dissociates a vpci from this connection	`A_DELVC ioctl`
`atm_allocate_bw`	Allocates constant bit rate bandwidth for this connection	`A_ALLOCBW ioctl`
`atm_allocate_cbr_bw`	Allocates constant bit rate bandwidth with more granularity than `atm_allocate_bw`	`A_ALLOCBW_CBR ioctl`
`atm_allocate_vbr_bw`	Allocates variable bit rate bandwidth	`A_ALLOCBW_VBR ioctl`
`atm_release_bw`	Releases previously allocated bandwidth	`A_RELSE_BW ioctl`

Note – The following discussion uses user space function names. Refer to TABLE A-4 for the corresponding kernel space function or `ioctl`.

To establish a data path, the application must first open the ATM driver and attach to a specific physical interface using `atm_open()` and `atm_attach()`. Next, the connection is associated with one or more VC(s), using `atm_add_vpci()`. If a call has been established using Q.2931 signalling, the vpci provided to `atm_add_vpci()` is the vpci that was included in the Q.2931 signalling messages received while establishing the call.

An encapsulation method must also be selected. The method of encapsulation is selected when the VC is associated with a stream (the `atm_add_vpci()` call). Currently, null and LLC encapsulation are supported. Null encapsulation implies that a message consists only of data preceded by a four-byte vpci. This type of encapsulation is most commonly used with raw mode. LLC encapsulation implies that an LLC header precedes the data. This header includes the SAP associated with the application's stream (using `atm_bind()`).

You can also select a mode of operation to determine the format of the message blocks passed to the ATM device driver. DLPI mode is set by default; however, the user can select raw mode with a call to `atm_setraw()`. DLPI mode implies that two or more mblocks will be sent to the driver. The first, which corresponds to the ctl buffer sent in the `putmsg()` system call, contains the `dlpi` message type, which is `dl_unitdata_req` for transmit and `dl_unitdata_ind` for receive. The vpci is included in this mblock as well. The `dl_unitdata_req` and `dl_unitdata_ind` header formats are defined in the header file `<sys/dlpi.h>`. The second and subsequent mblocks (corresponding to the data buffer in the `putmsg()` system call) contain the message.

Raw Mode Connections

Raw mode implies that the four-byte vpci is sent in the first mblock (corresponding to the ctl buffer in the `putmsg()` system call) followed by data. Any subsequent mblocks (the data buffer in the `putmsg()` system call) contain only data.

DLPI Mode Connections

Although the encapsulation and operational mode can be chosen independently, typically, DLPI mode is used for LLC-encapsulated traffic and raw mode is used for null encapsulation.

The driver's handling of packets depends on both the encapsulation method and the operational mode. For LLC-encapsulated traffic running in dlpi mode, the driver automatically adds the LLC header on transmit and strips the LLC header on receive before sending the message up the `dlpi` mode stream. In raw mode, however, the

driver does not modify the packets at all. This includes any header included with the packet. Thus, an application using raw mode and LLC encapsulation must include its own headers on transmit and receives data with the LLC header intact.

Received packets are directed to application streams based on the type of encapsulation in use. If a packet is null-encapsulated, it is sent up the stream associated with the vpci on which the packet was received. If a packet is LLC-encapsulated, it is sent to the stream that has bound (using `atm_bind()`) the SAP found in the LLC header.

APPENDIX **B**

Troubleshooting and Error Messages

Troubleshooting While Starting a SunATM Interface

There are many steps involved in making an interface active on an ATM network. Problems in your configuration may cause a failure at any number of points along the way. The following sections contain steps you can take to determine where in the process your system failed, and what to do to remedy the situation. If you continue to experience problems, information gathered from these steps will help your service provider diagnose the problem.

Generic Configuration

▼ To Diagnose Problems

1. **Make sure that there is an entry for the interface in `/etc/opt/SUNWconn/atm/atmconfig`.**

   Configuration of an interface begins during system boot. Configuration will be attempted for all interfaces listed in `/etc/opt/SUNWconn/atm/atmconfig`. For information about the format of this file, see “Editing the `/etc/opt/SUNWconn/atm/atmconfig` File” on page 23, and the `atmconfig(4)` man page.

2. **Check to see if any error messages were printed during the boot process.**

   If there were error messages, see “Error Messages” on page 83.

3. **Verify linkstate in** `qccstat(1M)`**.**

   This command indicates the signalling status of your interface. If the linkstate is not `DL_ACTIVE`, your interface is not communicating properly with your switch.

   - Make sure that your switch and interface are configured to run the same version of UNI signalling.

   The SunATM software supports UNI versions 3.0, 3.1, and 4.0; set the version for each interface in the `/etc/opt/SUNWconn/atm/atmconfig` file.

   - Verify that your interface is physically connected to the switch and that the switch sees the physical connection (most switches have a physical link LED for each port).

   If your interface is a multimode fiber interface, one possible cause for a bad physical connection is that transmit and receive are swapped. “transmit” on your interface should be connected to “receive” on the switch, and “receive” on your interface to “transmit” on the switch. There is generally writing on one of the cables in a transmit-receive pair so that the two cables are distinct.

4. **Verify that an address has been registered with the switch.**

   The `qccstat(1M)` command also lists all addresses registered to the interface with the switch. See “ATM Addresses and Address Registration” on page 46, for more information about address registration. If there are no addresses registered, the `ilmid` daemon on your system is not communicating properly with the switch.

   - Verify that there are incoming packets on VC 16 using `atmstat(1M)`.

   If there are no incoming packets, the switch is not responding to ILMI requests. Check its ILMI configuration.

   - Verify that there are outgoing packets on VC 16 using `atmstat(1M)`.

   If you do not see any outgoing packets on VC 16, your interface is not transmitting ILMI packets. Verify that `ilmid` is running on your system, and if necessary, start it in the background. Starting `ilmid` with the `-v` flag causes it to print a notice for every message received or transmitted, along with other diagnostic information.

5. **Interfaces that are not running Classical IP or LAN Emulation will not appear in the output of the** `ifconfig` **command.**

   `ifconfig(1M)` displays interfaces that have been configured for IP. In order to support IP, ATM interfaces must run either Classical IP or LAN Emulation. Therefore, an ATM interface that is not configured to support IP by running one of these two protocols will not be displayed by `ifconfig`.

6. **Verify the packets that are moving over the network with the** `/etc/opt/SUNWconn/bin/atmsnoop` **command.**

Classical IP Configuration

1. **Check all of the generic configuration points.**

   These are issues that apply to all SunATM interfaces, so they all must be working in order for Classical IP to work.

2. **Verify the output of** `ifconfig(1M)`**.**

   Executing the command `ifconfig -a` displays the SunATM interface, `ba`*N*, where *N* is the instance number.

   - If your interface does not appear, an error probably occurred during the boot process.

   Check for error messages during the boot process. The meanings and possible solutions for error messages can be found in "Error Messages" on page 83.

   - If your interface appears, but has incorrect information, verify your configuration files.

   The information given to `ifconfig` comes from the `/etc/opt/SUNWconn/atm/atmconfig` and `/etc/opt/SUNWconn/atm/aarconfig` files. Check the entries in those files that apply to this interface and verify their contents. For descriptions of the file formats, see "Editing the `/etc/opt/SUNWconn/atm/atmconfig` File" on page 23, and "Editing the `/etc/opt/SUNWconn/atm/aarconfig` File" on page 26, or the `atmconfig(4)` and `aarconfig(4)`man pages.

3. **Check the** `setup_state` **with** `aarstat(1M)`**.**

   This command will provide information about the Classical IP status on your interface. The `setup_state` refers to the completion of the `aarsetup` program.

   - If the `setup_state` is setup-started, it indicates that the `aarsetup` program has not completed; it may be delayed by slow switch responses, or failed attempts to register ATM addresses in `/etc/opt/SUNWconn/atm/aarconfig`. Make sure that the local address given for your interface in `/etc/opt/SUNWconn/atm/aarconfig` is unique to this switch. Using `$myaddress` and the reserved server addresses is a good way to guarantee that all addresses are unique. After making any changes to `/etc/opt/SUNWconn/atm/aarconfig`, run `aarsetup` again.
   - If the state is not setup-started or setup-finished, verify that the addresses and interfaces in `/etc/opt/SUNWconn/atm/aarconfig` are valid, and run `aarsetup` again. If you see any error messages, check their meaning in "Error Messages" on page 83.

4. **Verify the** `interface_state` **in** `aarstat(1M)`**.**

   The `interface_state` is either up or down, and reflects the linkstate given in the output of `qccstat`. If the linkstate is `DL_ACTIVE`, the `interface_state` is up; otherwise, the `interface_ state` is down. If `aarstat` indicates that the `interface_state` is down, try the suggestions for a linkstate that is not `DL_ACTIVE`, given in "Generic Configuration" on page 73.

5. **Make sure Classical IP is configured correctly.**

   The `aarstat(1M)` output lists several parameters for Classical IP. The field `arpcsmode` lists whether Classical IP is running as a client, a server, or stand-alone (a client with no server configured). Verify that this is correct; if it is not, check your `/etc/opt/SUNWconn/atm/aarconfig` file entries.

6. **If the system is a Classical IP client, verify the server connection.**

   On systems running in client mode, `aarstat` also provides information about the server. Verify the server address, and that the `server_state` is connected.

7. **If the `server_state` is no-connection or connecting.**

   The system is likely having a problem establishing a connection to the server. Verify that the server address is correct, and that there is a system on the network which has registered that address. The server and applicable switch ports must also be configured to support UNI signalling, also called Q.2931 or Q.93b.

8. **Verify that addresses are resolved and connections are made with the `ping(1M)` command.**

   Once you have two systems configured and running to this point, they should be able to `ping` each other. To ping client2 from client1:

```
% ping client2
client2 is alive
```

   If the `ping` is not successful:

   1. Check that ARP requests are being sent to the server.

   Find the `server_vci` in the output of `aarstat`. Then run `atmstat`, and verify that there are outgoing packets on that VC. If not, make sure that your interface is up and configured properly.

   2. Make sure that you are receiving ARP responses from the server.

   In the `atmstat` output, check the output packets for the server VC (found in the `aarstat` information). If none are being received, your server is not responding to ARP requests from the client. If it is a SunATM server, verify its Classical IP status with the suggestions given here. If not, verify that it is up and running as a server.

   3. Make sure the address is resolved correctly.

   Run the `atmarp` command for the system you are trying to `ping`, and verify that its IP address has been resolved to the correct ATM address. If not, make sure that the remote system is registering the correct address with the ATM ARP server. If the address has not been resolved at all, make sure that the remote system has a connection to the server.

4. Verify that a connection has been established between the two systems.

The output of `qccstat` lists the source and destination addresses of all open connections. You should have at least one connection to the server, and you should also see a connection to the remote host you are trying to `ping`. If not, make sure both interfaces are up and registered with the switch, and that both interfaces and the switch are running UNI signalling (Q.2931 or Q.93b).

5. Check for IP problems.

If the address has been resolved correctly, and a connection has been established between the two systems, but they still cannot `ping`, the problem is likely outside the scope of ATM.

LAN Emulation Configuration

1. **Check all of the generic configuration points.**

   These are issues that apply to all SunATM interfaces, so they must all be working in order for LAN Emulation to work.

2. **Verify the output of** `ifconfig(1M)`**.**

   Executing the command `ifconfig -a` should display the ATM LAN Emulation interface, `lane`*N*, where *N* is the instance number.

   - If your interface does not appear, an error probably occurred during the boot process.

   Check for error messages during the boot process. The meanings and possible solutions for error messages can be found in “Error Messages” on page 83.

   - If your interface appears, but has incorrect information, verify your configuration files.

   The information given to `ifconfig` comes from the `/etc/opt/SUNWconn/atm/atmconfig` and `/etc/opt/SUNWconn/atm/laneconfig` files. Check the entries in those files that apply to this interface and verify their contents. For descriptions of the file formats, see “Editing the `/etc/opt/SUNWconn/atm/atmconfig` File” on page 23 and “Editing the `/etc/opt/SUNWconn/atm/laneconfig` File” on page 35, or the `atmconfig(4)` and `laneconfig(4)` man pages.

3. **Check the** `setup_state` **with** `lanestat(1M)`**.**

   This command provides information about the LAN Emulation status on your interface. The `setup_state` refers to the completion of the `lanesetup` program.

   - If the `setup_state` is setup-started:

This indicates that the `lanesetup` program has not completed; it may be delayed by slow switch responses, or failed attempts to register ATM addresses in `/etc/opt/SUNWconn/atm/laneconfig`. Make sure that the local address given for your interface in `/etc/opt/SUNWconn/atm/laneconfig` is unique to this switch. Using the variable `$myaddress` for all systems is a good way to guarantee that all addresses are unique. After making any changes to `/etc/opt/SUNWconn/atm/laneconfig`, run `lanesetup` again.

- If the state is not setup-started or setup-finished:

Verify that the addresses and interfaces in `/etc/opt/SUNWconn/atm/laneconfig` are valid, and re-run `lanesetup`. If you see any error messages, check their meanings in "Error Messages" on page 83.

4. **Verify that a connection has been made to the LAN Emulation server (LES).**

   A LAN Emulation client must establish and maintain a connection to the LES. In most cases, the LES also establishes and maintains a second connection to the client. Find the LES address in the output of `lanestat`, and then look for connections with that address as the destination or source in the output of `qccstat`.

   If you do not see any connections with that address, take the appropriate action from the list below:

   - If you have a LAN Emulation configuration server (LECS):

   Make sure that the correct address is configured for the LECS. By default, the SunATM software uses the ATM Forum well-known address. If your LECS uses a different address, enter the alternate address in the `/etc/opt/SUNWconn/atm/laneconfig` file. See "Editing the `/etc/opt/SUNWconn/atm/laneconfig` File" on page 35, for information on editing `/etc/opt/SUNWconn/atm/laneconfig`. You can check the address currently being used in the output of `lanestat`.

   - If you do not have an LECS:

   One of the LECS functions is to provide the LES address, so if you do not have an LECS, you must provide the address. Create an entry in `/etc/opt/SUNWconn/atm/laneconfig`. See "Editing the `/etc/opt/SUNWconn/atm/laneconfig` File" on page 35. You can check the LES address currently being used in the output of `lanestat`.

   - Verify that the LECS, if present, and LES are configured properly.

5. **Verify that a connection has been made to the BUS.**

   In addition to the LES connection(s), a LAN Emulation client must also establish and maintain a connection to the BUS, and the BUS typically establishes and maintains a second connection to the client. You can find the BUS ATM address in the output of `lanestat`, and then verify that there is a connection with that address as the destination, and probably a second connection with that address as source, in the output of `qccstat`. If there are not any connections, verify that the BUS is configured properly.

6. **Verify that the host has joined the Emulated LAN.**

   The `lanestate` field in the output of `lanestat` indicates that the client is in the active state.

   If your system cannot join the emulated LAN, there may be a problem with the way in which your LAN Emulation Services are configured. If the Emulated LAN uses an MTU size larger than 9 Kbytes, the SunATM host will not join (9 Kbytes is the largest MTU size supported by the SunATM product). If the host is not able to join, an error message will be printed with an explanation.

7. **Verify that addresses are resolved and connections are made with the `ping` command.**

   Once you have two systems configured and running to this point, they should be able to `ping` each other. To ping client2 from client1:

```
% ping client2
client2 is alive
```

   If the `ping` is not successful:

   1. Check that the IP hostname or address is resolved to a MAC address.

   LAN Emulation requires two address resolution steps to make a call. The first is to resolve an IP address to a MAC address. From the perspective of IP and ARP, this works exactly as it does on an Ethernet interface; using the `arp` command, you can verify that this resolution has been made correctly. If it has not, verify the connections to the BUS, and make sure data is being transmitted and received on the connection(s) to the BUS by finding the VC in the output of `qccstat`, and looking at the statistics for that VC in `atmstat`.

   2. Check that the MAC address has been resolved to an ATM address.

   This is the second address resolution step, and is accomplished by the LAN Emulation software and communication with the LES. You can use the `lanearp` command to verify that MAC addresses have been properly resolved to ATM addresses. If they have not, verify the connections to the LES, and make sure data is being transmitted and received on the connection(s) to the LES by finding the VC in the output of `qccstat` and looking at the statistics for that VC in `atmstat`.

   3. Verify that a connection has been established between the two systems.

   The output of `qccstat` lists the source and destination addresses of all open connections. There you should see a connection to the remote host you are trying to `ping`. If not, make sure both interfaces are `up` and registered with the switch, and that both interfaces and the switch are running UNI signalling (Q.2931 or Q.93b).

   4. Check for IP problems.

If the address has been resolved correctly, and a connection has been established between the two systems, but they still cannot `ping`, the problem is likely outside the scope of ATM.

Common Problems

This section describes some common problems that you may experience during or after the SunATM adapter installation. Please review this section before calling Sun Service for assistance.

Are you replacing an old SunATM SBus adapter?

1. **If you are replacing an old SunATM/S 155 adapter with a new adapter, you must edit the `/etc/path_to_inst` file to remove the old device instance.**

   The SunATM/S 155 adapter originally shipped with an FCode name of "`ba`" (part numbers 501-2794-07, 501-2795-05, and prior versions). Since then, Sun Microsystems, Inc. has changed the naming convention to include `SUNW` at the beginning of every device name. When a third-party adapter was found, which also used the name property "`ba`", the SunATM/S 155 adapter was updated to use the "`SUNW,ba`" name property instead (the change was made to part numbers 501-2794-08, 501-2795-06, and compatible versions).

   As a result, when an older SunATM/S 155 adapter (with the "`ba`" name property) is replaced by a newer SunATM/S 155 adapter (with the "`SUNW,ba`" name property), the system does not recognize the new adapter as a replacement. Instead, the system sees it as a new interface and assigns a new instance number to the adapter. The `/etc/path_to_inst` file is created by the Solaris operating environment to identify installed devices and their instance numbers. When a SunATM/S 155 adapter (with the "`ba`" name) is installed in a system, `/etc/path_to_inst` has an entry, similar to the following, to identify it as `ba0`:

```
"/sbus@1f,0/ba@0,0" 0 "ba"
```

   When a replacement adapter (with the "`SUNW,ba`" name) is installed into the same location and the system is rebooted, it treated as a new device and a new entry in `/etc/path_to_inst` is created for `ba1`:

```
"/sbus@1f,0/SUNW,ba@0,0" 1 "ba"
```

To correct this, delete the original entry that contains the “ba” name. Then, modify the second field of the new entry, which contains the name “SUNW,ba”, to reflect the proper instance number. In this example, the new entry is designated as instance 0:

```
“/sbus@1f,0/SUNW,ba@0,0” 0 “ba”
```

After you have modified and saved /etc/path_to_inst, reboot the system for the changes to take effect.

Note – The physical name listed in /etc/path_to_inst varies from one architecture to another and might not match the previous examples exactly. However, modify only the instance number field. Be sure to leave all other fields as they are.

Are you trying to use the /usr/sbin/arp command?

Since the Classical Internet Protocol (IP) network model resolves IP-to-ATM address pairs rather than IP to MAC address pairs, the /usr/sbin/arp command does not support Classical IP interfaces at this time. A version of the arp command, /etc/opt/SUNWconn/atm/bin/atmarp, provides similar functionality for Classical IP interfaces. Refer to the atmarp (1M) man page for more information.

Are you using a Router with Classical IP and LAN Emulation (LANE)?

Performance problems occur if a router uses ATM Classical IP (default 9180 byte MTU) and LAN Emulation (default 1500 byte MTU) links simultaneously when a TCP connection is set up using one interface in one direction and the other interface in the opposite direction, TCP is confused about the maximum packet size.

For example, suppose a TCP connection is set up between Host A and Host B, where packets from Host A travel to Host B over the LANE interface and packets from Host B travel to Host A travel over the Classical IP interface. Host A attempts to send a 9180 byte packet that cannot traverse the LANE network to Host B. TCP recovers from this error and retransmits the packet, but a significant performance loss will be noted.

Possible workarounds to improve performance are:

- Adjust the MTU size, if possible, of the Classical IP link to 1500 bytes.
- Depending upon the network topology, adjust the routing table on Host B to ensure that the route back to Host A points to the LANE interface.

This problem is not unique to ATM networks. It may affect any network configuration that has multiple routes with differing MTUs (such as FDDI and Ethernet or Token Ring). The problem is more pronounced with ATM subnets because of the different default MTUs of Classical IP and LANE.

Are you trying to use the `/usr/sbin/snoop` command?

The `/usr/bin/snoop` command, which can be used to detect network problems, does not support SunATM interfaces at this time. A version of the snoop command, `/etc/opt/SUNWconn/atm/bin/atmsnoop`, provides this support. Refer to the `atmsnoop(1M)` man page for more information.

Do you want to increase system performance by adjusting TCP/IP parameters?

TCP/IP performance over an ATM network can be poor unless you carefully configure your network. Poor performance usually occurs because the TCP/IP packets are segmented into cells for transmission by the ATM software. Therefore, a loss of a single cell can cause the loss of an entire TCP/IP packet which can lead to retransmissions that congest the network. When it detects congestion, the destination system reduces the transmission rate, which significantly reduces the network performance.

You can achieve better network performance from the SunATM adapter and software by adjusting your application's socket buffer size to 48 Kbytes. Refer to the application's documentation for instructions on how to set the socket buffer size.

Are you trying to mount a diskless, dataless, or autoclient system?

The SunATM adapters do not currently support diskless, dataless, or autoclient systems. The `root` filesystem must be local for the SunATM adapter to operate.

Did the `atmtest` diagnostic fail?

If the bandwidth or outstanding packets value is set too high on your system, the SunVTS `atmtest` diagnostic can fail, giving a error similar to the following:

```
SUNWvts.atmtest.4000 09/17/98 17:33:10 atmtest ba0
WARNING: “VC30 dropped pkt, seq: exp=41, obs=43; len: exp=1747,
obs=6022”
```

To correct this error, reduce the bandwidth or the number of outstanding packets in the SunVTS `atmtest`.

Error Messages

This section includes a list of some of the most common error messages you might see while configuring and bringing up your SunATM interface. For each message, there is a brief explanation of the problem and a possible solution.

Error Messages from `S00sunatm`

`Cannot find ATM utilities in /etc/opt/SUNWconn/atm/bin; exiting S00sunatm.`

The SunATM utility directory `/etc/opt/SUNWconn/atm/bin` does not exist. Make sure that the SUNWatm package installation completed successfully (see "Checking the Package Installation Using `pkginfo`" on page 4, for more information). You might need to reinstall the package.

`Cannot find atmconfig file in /etc; exiting S00sunatm.`

The `/etc/opt/SUNWconn/atm/atmconfig` file provides configuration information to the `S00sunatm` script so that it can bring up the SunATM interfaces during system boot. If the `/etc/opt/SUNWconn/atm/atmconfig` file is not present, `S00sunatm` prints this warning message and exits. The `/etc/opt/SUNWconn/atm/atmconfig` file is installed with the `SUNWatm` package as `/etc/opt/SUNWconn/atm/atmconfig.template`; if you choose `autoconfiguration` or if no previous `/etc/opt/SUNWconn/atm/atmconfig` file exists, `pkgadd` copies this template file to `/etc/opt/SUNWconn/atm/atmconfig`. If a previous `/etc/opt/SUNWconn/atm/atmconfig` file exists, it is not overwritten. See "Editing the `/etc/opt/SUNWconn/atm/atmconfig` File" on page 23.

`warning: can't plumb <device>; no UNI version provided`

The first entry in `/etc/opt/SUNWconn/atm/atmconfig` for a physical interface must include a UNI value in the second field.

`warning: can't plumb <uni version> on <device>; <uni version> already plumbed`

The system encountered an entry which attempted to plumb a signalling version on an interface that has already been plumbed with a different signalling version. The script ignores the new UNI version and continues processing the entry and the remaining entries in the file.

`warning: can't plumb <lane instance>: too many lane instances on <device>`

A physical interface will support up to *n* lane instances, where *n* is the number of MAC addresses on the board (or 1 if the board has no MAC address).You can check the number of MAC addresses on a board using the `count` option of the `atmgetmac(1m)` command. If an entry is encountered that attempts to plumb more LANE instances than allowed, this message occurs; processing will continue with the next entry in the file.

`warning: can't plumb signalling on <device>`
`warning: can't plumb classical IP interface <device>`
`warning: can't plumb <lane instance> on <device>`

An error occurred when the script attempted to run `atmplumb(1m)` (either to plumb signalling, classical IP, or LAN Emulation on an interface) with information specified in `/etc/opt/SUNWconn/atm/atmconfig`. The `atmplumb` program will generally display an error message indicating why it failed; use that information to check your values in the `/etc/opt/SUNWconn/atm/atmconfig` entry for device. The script proceeds to read and process the remaining entries in `/etc/opt/SUNWconn/atm/atmconfig`, although further entries for the failed interface are not processed correctly.

`warning: invalid interface <lane instance>`

The minor number provided in a logical interface name was not in the range 0 - 255. The script proceeds without attempting to configure the invalid lane device.

`warning: only one classical ip hostname is allowed on <device>`

An additional entry was found containing a Classical IP hostname after an initial Classical IP hostname was already plumbed for the given device. Multiple Classical IP instances are not supported on a single physical interface. The script ignores additional Classical IP information for a physical interface.

`warning: <laneN> entry must appear before <laneN:X> entry`

When you use logical interface names, the first entry in `/etc/opt/SUNWconn/atm/atmconfig` must always be either `lane`*N* or `lane`*N*`:0`, which are equivalent. All entries that appear before the `lane`*N* or `lane`*N*`:0` entry are ignored.

`Please install <SUNWatm>`

A required software package is not installed on the system. Install the package and reboot the system.

`warning: extra fields for <device> will be ignored`

There were additional fields in the `/etc/opt/SUNWconn/atm/atmconfig` entry for the given device name. The script proceeds, ignoring the additional fields.

`warning: duplicate entry <lane device>`

There were multiple entries in `/etc/opt/SUNWconn/atm/atmconfig` using the same LAN Emulation instance number. This is not a fatal error; the script continues to run. However, only the first entry for each LAN Emulation instance number is configured for LAN Emulation.

`warning: not enough fields to configure <device>`

The `/etc/opt/SUNWconn/atm/atmconfig` entry for the given device did not have all the required fields. You must edit the `/etc/opt/SUNWconn/atm/atmconfig` file (see “Editing the `/etc/opt/SUNWconn/atm/atmconfig` File” on page 23), filling in all the appropriate information, and reboot the system. Empty fields should be indicated with a hyphen (`-`).

`warning: ifconfig failed for classical IP interface <device>`
`warning: ifconfig failed for <lane instance>`

The script attempted to run `ifconfig` for the specified interface. Error messages indicate why `ifconfig` failed; use that information to check your values in `/etc/opt/SUNWconn/atm/atmconfig`. In particular, verify that the hostname you provide in `/etc/opt/SUNWconn/atm/atmconfig` appears in the `/etc/hosts` file on your system.

`warning: invalid lane instance (<lane instance>) for <device>`

The lane instance number provided in `/etc/opt/SUNWconn/atm/atmconfig` was not in the range 0 to 999. The script proceeds without attempting to configure the invalid lane instance.

`warning: aarsetup failed; could not configure classical IP interfaces`
`warning: lanesetup failed; could not configure LAN Emulation interfaces`

Either the LAN Emulation or the Classical IP startup script failed and exited with an error value. Check the error messages that were printed by `aarsetup` or `lanesetup`, and verify the values you have entered in `/etc/opt/SUNWconn/atm/aarconfig` and/or `/etc/opt/SUNWconn/atm/laneconfig`.

Error Messages from `aarsetup` and `lanesetup`

`aarsetup: could not become control process`
`lanesetup: could not become control process`

An instance of the setup program was running when another instance was started up. The second instance exits with this error message. Make sure that there is not a previous instance of the program still running. The setup program might take a while to complete if the switch is slow to respond.

`aarsetup: could not open stream to Q93B`
`lanesetup: could not open stream to Q93B`

The program was unable to communicate with the Q93B driver. Make sure that you run `aarsetup` or `lanesetup` as root, and that the SUNWatm package has been properly installed.

`aarsetup: could not scan input file`
`lanesetup: could not scan input file`

The program was unable to open the `/etc/opt/SUNWconn/atm/aarconfig` or `/etc/opt/SUNWconn/atm/laneconfig` file (or the file specified on the command line). Verify that the appropriate file exists, and has the proper permissions. Also make sure you run `aarsetup` or `lanesetup` as root.

`aarsetup: exiting because of errors`
`lanesetup: exiting because of errors`

Errors were encountered while parsing the `/etc/opt/SUNWconn/atm/aarconfig` or `/etc/opt/SUNWconn/atm/laneconfig` file, so the setup program cannot successfully complete. Correct the error condition and then execute either `aarsetup` or `lanesetup`.

`aarsetup: <interface> running as a server, but PVC-only ‘t’ entries exist`

The `aarsetup` program has found an *L* entry in `/etc/opt/SUNWconn/atm/aarconfig`, meaning that this interface will be running as a server; however, there are table entries (*t* entries) containing only PVCs, which cannot be entered into the server’s ATM ARP table. Verify your interface’s status (server, client, or stand-alone), make sure all *t* entries include ATM addresses, and execute `aarsetup`. See “Editing the `/etc/opt/SUNWconn/atm/aarconfig` File” on page 26, for more information.

`aarsetup: waiting for ilmid to provide prefix`
`lanesetup: waiting for ilmid to provide prefix`

In some cases, the address registration process may take several minutes. If so, `aarsetup` or `lanesetup` prints out this message saying that it cannot complete until address registration completes. If the messages continue for more than a minute or two, verify your connection to the switch, and that the switch and interface are both supporting ILMI.

`undefined variable`

You used a variable in a configuration file without using a set statement to assign the value. Add a `set` statement, or correct the variable name, and run `aarsetup` or `lanesetup` again. See "Using Variables in the `/etc/opt/SUNWconn/atm/aarconfig` File" on page 29, and "Using Variables in the `/etc/opt/SUNWconn/atm/laneconfig` File" on page 38, for more information.

`variable already defined`

You tried to set a variable that had been previously set in the same configuration file. Remove the second assignment and run `aarsetup` or `lanesetup` again.

`variable name ill-formed`

You created a variable in `/etc/opt/SUNWconn/atm/aarconfig` or `/etc/opt/SUNWconn/atm/laneconfig` that was syntactically invalid. Variable names are a combination of letters, digits, and underscores (_). Choose a conforming variable name and run `aarsetup` or `lanesetup` again.

`variable name too long`

You created a variable in `/etc/opt/SUNWconn/atm/aarconfig` or `/etc/opt/SUNWconn/atm/laneconfig` with a name that is greater than the maximum length (32 characters). Choose a variable name shorter than 32 characters and run `aarsetup` or `lanesetup` again.

`variable value too long`

You assigned a value longer than the maximum value length of 128 characters to a variable in a configuration file. If you want a longer value, use a combination of variable names, with each value less than 128 characters. After correcting the variable value lengths, run `aarsetup` or `lanesetup` again.

`ifname:cannot join ELAN (frame size too large; please use a different ELAN and rerun lanesetup)`

The largest MTU size supported by the SunATM software is 9 kilobytes. If the LAN Emulation Services try to set a size larger than 9 Kbytes, the SunATM client cannot join the emulated LAN. Reset your LAN Emulation services to use an MTU size less than or equal to 9 Kbytes, and rerun `lanesetup` to join the emulated LAN.

`ifname: frame-size change (please rerun lanesetup)`

The MTU size was changed by the LAN Emulation Services; rerun `lanesetup` to notify IP of the change. There is a slight chance that TCP connections will remain open during this change, and if that is the case, performance on those connections is affected by the change. Either restart the affected applications or reboot the system if this becomes a problem.

`<ifname> could not download the MAC address`

This message indicates that an error occurred while `lanesetup` was attempting to retrieve a MAC address for the indicated interface. Most likely the kernel is out of memory or you have not run `atmplumb` for the specified interface.

`Could not find driver for <ifname>`

Each LAN Emulation interface is associated with an ATM driver when LAN Emulation is set up by `atmplumb`. This message indicates that this interface/driver association has not been made, most likely because you have not run `atmplumb` for the specified interface.

`Not enough MAC addresses on <ATM interface>`

The number of Emulated LANs that can be joined over a single physical interface is limited by the number of MAC addresses on the ATM interface board. This message indicates that you tried to join more Emulated LANs than allowed by the number of MAC addresses on the specified interface. You can find the number of MAC addresses on an interface by using the `count` option on the `atmgetmac(1M)` command; the number of Emulated LANs and lane instances indicated in `/etc/opt/SUNWconn/atm/atmconfig` and `/etc/opt/SUNWconn/atm/laneconfig` should not exceed this number. See "Supporting Multiple Emulated LANs on a Single Interface" on page 57.

Error Messages from the Kernel Drivers

`q93b: warning: link coming back up on <interface>, but ilmid is not running`

The link has gone down and come back up on an interface, but `ilmid` is not running at this time. Register addresses with the switch again, because both the interface and switch must clear out their address tables when the link goes down. Start `ilmid`; if the interface does not seem to be running properly after doing this, you may need to reboot the system. It is likely that the interface was in an unusual or unknown state when the link came back up, and may need to be taken down completely by rebooting.

Index

B

C

D

E

F

H

I

K

L

M

N

P

Q

R

S

T

U

V