IRIS® ATM Configuration Guide
SG–2236 2.3

Document Number 007–2333–005

CONTRIBUTORS:

Written and illustrated by Carlin Otto

Revised by Julie Boney

Engineering contributions by Irene Kuffel, Jean-Michel Pittet, and Thomas Skibo

New Features

IRIS® ATM Configuration Guide SG–2236 2.3

This rewrite of the IRIS ATM Configuration Guide documents ATM release 2.3 and supports the 6.5 release of the IRIX operating system.

Record of Revision

Version	*Description*
2.3	June 1998 Original rewrite to support the IRIX 6.5 operating system

Contents

Page

Page

Page

Page

Page

Figures

Page

Page

Tables

Page

About This Guide

This publication documents IRIS Asynchronous Transfer Mode (ATM) release 2.3 and supports the IRIX 6.5 operating system running on the Origin series and the Onyx2, CHALLENGE, and Onyx platforms.

The IRIS ATM product is hardware and software that together allow applications to transmit and receive data over an ATM connection. IRIS ATM is an excellent data communication or networking solution for applications that require high-speed, constant, or nearly constant data rates.

Related Publications

The following documents contain additional information that might be helpful:

- *IRIS ATM-OC3 Board for CHALLENGE and Onyx Installation Instructions*, publication 108-0113-xxx
- *IRIS ATM-OC3c 4Port XIO Board Installation Instructions for Origin2000 and Onyx2*, publication 108–0159–xxx
- *IRIS ATM API Programmer's Guide*, publication 007–2334–*xxx*

Ordering Publications

Silicon Graphics maintains publications information at the following URL:

`http://techpubs.sgi.com/library`

The preceding website contains information that allows you to browse documents online, order documents, and send feedback to Silicon Graphics.

The *User Publications Catalog*, publication CP–0099, describes the availability and content of all Cray Research hardware and software documents that are available to customers. Cray Research customers who subscribe to the Cray Inform (CRInform) program can access this information on the CRInform system.

Cray Research also has documents available online at the following URL:

`http://www.cray.com/swpubs`

To order a Cray Research or Silicon Graphics document, either call the Distribution Center in Mendota Heights, Minnesota, at +1–612–683–5907, or send a facsimile of your request to fax number +1–612–452–0141.

Cray Research employees may send their orders via electronic mail to `orderdsk` (UNIX system users).

Customers outside of the United States and Canada should contact their local service organization for ordering and documentation information.

Acronyms Used in This Guide

The following acronyms are used throughout this guide:

ATM	Asynchronous Transfer Mode
ATMARP	ATM address resolution protocol as specified in RFC 1577
BLLI	broadband low-layer information
CBR	constant bit rate
CSPDU	AAL5 convergence sublayer protocol data unit
ILMI	interim local management interface
LIS	logical IP subnetwork as defined in RFC 1577
PVC	permanent virtual circuit
SONET	synchronous optical network
SVC	switched virtual circuit
UNI	ATM user-network interface
VBR	variable bit rate
VC	virtual channel

Conventions

The following conventions are used throughout this document:

Convention	Meaning
`command`	This fixed-space font denotes literal items such as commands, files, routines, path names, signals, messages, and programming language structures.

variable	Italic typeface denotes variable entries and words or concepts being defined.
`user input`	This bold, fixed-space font denotes literal items that the user enters in interactive sessions. Output is shown in nonbold, fixed-space font.
[ ]	Brackets enclose optional portions of a command or directive line.
...	Ellipses indicate that a preceding element can be repeated.

Product Support

Silicon Graphics provides a comprehensive product support and maintenance program for its products. If you are in North America and would like support for your Silicon Graphics supported products, contact the Technical Assistance Center at 1-800-800-4SGI. If you are outside North America, contact the Silicon Graphics subsidiary or authorized distributor in your country.

Reader Comments

If you have comments about the technical accuracy, content, or organization of this document, please tell us. You can contact us in any of the following ways:

- Send us electronic mail at the following address:

  `techpubs@sgi.com`

- Contact your customer service representative and ask that an SPR or PV be filed. If filing an SPR, use `PUBLICATIONS` for the group name, `PUBS` for the command, and `NO-LICENSE` for the release name.

- Call our Software Publications Group in Eagan, Minnesota, through the Customer Service Call Center, using either of the following numbers:

  1–800–950–2729 (toll free from the United States and Canada)

  +1–612–683–5600

- Send a facsimile of your comments to the attention of "Software Publications Group" in Eagan, Minnesota, at fax number +1–612–683–5599.

We value your comments and will respond to them promptly.

Overview of IRIS ATM [1]

This chapter provides an overview of IRIS ATM and the protocols upon which IRIS ATM is based.

1.1 What Are ATM and SONET?

The *synchronous optic network (SONET)* protocol is a physical layer communication technology, supporting transmission speeds such as 51.84 Mbit/s, 155.52 Mbit/s, 622.08 Mbit/s, and 2.488 gigabits per second. The *asynchronous transfer mode (ATM)* protocol is a data link and network layer switching protocol that supports almost any bit rate. SONET defines the manner in which data is encoded and transported over the line (that is, the fiber optic connection). ATM defines the manner in which the data is grouped into packets and how it is routed from endpoint to endpoint. ATM handles very small-sized cells in a manner that allows simultaneous transmission of multiple data streams at different rates. The streams can be different types of digitized data (for example, audio, video, and text).

Unlike today's popular network protocols (for example, Ethernet, FDDI, and Token Ring), ATM supports applications that require a steady, constant flow of data. Video applications (for example, teleconferencing, video on demand, and real-time long-distance imaging) are some of the main markets for this communication technology because the human eye and ear are highly sensitive to variations in time delays and synchronizing of visual data and sound. Table 1, page 2, summarizes some of the major differences between the common network technologies currently in use and the ATM technology.

Table 1. Comparison of ATM and Legacy Network Technologies

Today's common network technologies	ATM
Shared medium: Each station has exclusive access to the shared network medium for a short length of time, then releases the medium to allow other stations access.	Dedicated link: Each station has exclusive access all the time to its communication medium (the physical link).
Single stream of data: During a station's access, only one data stream is transmitted.	Multiple streams of data: Multiple data streams (virtual channels) can be simultaneously transmitted over a single physical connection.
Variable spacing between packets: Access times for transmission on the network medium are not predictable. Variable spacing between PDU arrival times are inherent in the design.[1]	Guaranteed, fixed spacing between packets: Each data stream can be guaranteed to have predictable, extremely reliably–spaced access to the communication medium. That is, the medium supports constant bit rate, in addition to more variable services.
Design inherently supports broadcasting, since every station sees every PDU.	Does not easily support broadcasting since each PDU is seen only by the two endpoints involved in the data stream and, in some cases, the switches between them.
Single transmission rate: Transmission rate is static at the network medium's built-in rate.	Multiple simultaneous transmission rates: Each data stream can specify its own transmission rate.

ATM allows each user to describe the data flow characteristics (that is, the *traffic contract*) wanted from the ATM connection. Some of the currently defined types of data flows are as follows:

1 PDU = protocol data unit, which is a frame, packet, or cell, depending on the technology's terminology.

- A steady, constant flow, called *constant bit rate (CBR)*, sometimes referred to as circuit emulation. The flow is specified as occurring at an absolutely steady or peak rate.
- A guaranteed, although fluctuating flow, called *variable bit rate (VBR)*. The flow is controlled by three parameters: a peak cell rate (the maximum rate that can ever be used on the VC), a sustainable rate (the average rate over time), and a maximum burst size.
- A flow that guarantees delivery, but does not conform to a timely delivery schedule, referred to as *available bit rate (ABR)*. [2]
- A flow that does not guarantee conformance to any specific performance parameters and, in fact, does not even guarantee delivery, referred to as *unspecified bit rate (UBR)* or best effort.

ATM defines the data link control and network layers, as illustrated in Figure 1, page 3, and is commonly implemented over a synchronous optical network (SONET) physical layer. Other network layers (such as IP) tunnel through the ATM network by *encapsulation*. ATM and SONET are each described briefly in the paragraphs that follow.

<table>
<tr><td>Transport Layer</td><td colspan="2">For example, TCP, UDP</td></tr>
<tr><td>Overlayered Network Protocols</td><td colspan="2">For example, IP encapsulated within ("tunneled through") ATM</td></tr>
<tr><td>Network Layer</td><td colspan="2">ATM addressing and routing</td></tr>
<tr><td rowspan="3">Data Link Control Layer</td><td rowspan="2">ATM Adaptation Layer (AAL)</td><td>Convergence</td></tr>
<tr><td>Segmentation and Reassembly</td></tr>
<tr><td colspan="2">ATM</td></tr>
<tr><td rowspan="2">Physical Layer</td><td rowspan="2">SONET</td><td>Transmission Convergence Sublayer</td></tr>
<tr><td>Physical Medium Sublayer</td></tr>
</table>

a11488

Figure 1. ATM within the OSI Protocol Stack

2 IRIS ATM does not currently support ABR.

1.1.1 ATM

ATM is a connection-oriented, packet-based protocol that allows multiple logical data streams (for example, different videos) to be transmitted simultaneously over a single physical connection. The ATM driver passes data from multiple applications to the ATM hardware where the data streams are stored as separate queues. The data is segmented into *ATM cells* and *multiplexed* into a single physical stream (see Figure 2, page 5). Each ATM cell is 53 bytes long, of which 5 bytes are ATM overhead and 48 bytes are upper-layer data (also known as payload).

Each logical data stream is called a *virtual channel* (VC). All of the VCs from a transmitting endpoint can share one physical link to the ATM switch. However, at the switch, the VCs might be routed onto different outgoing physical links, depending on their final destinations. In this way, they can follow their *virtual channel connection (VCC)* to the destination endpoint. ATM requires that the endpoint-to-endpoint physical connection be established before transmission occurs for a VC's first bit of data.

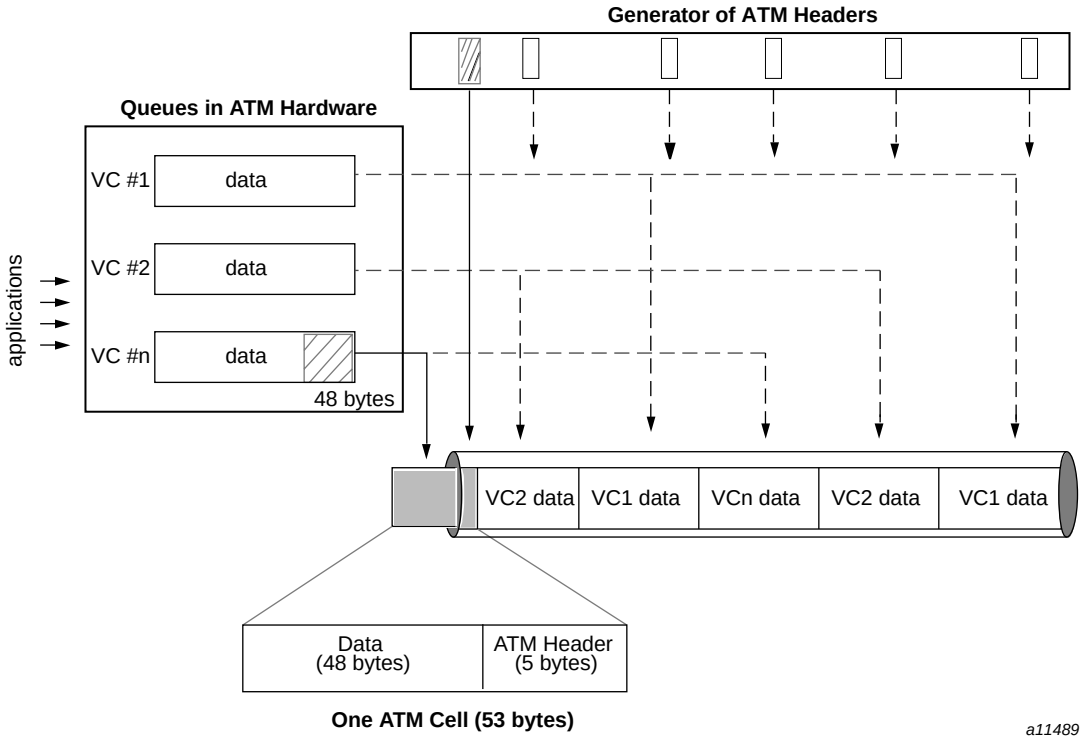


Figure 2. Segmenting and Multiplexing Data from Multiple VCs

When each VC is set up, the user specifies a transmission rate, an upper layer conversion protocol (referred to as the *ATM adaptation layer (AAL)*), and, for some implementations (for example, switched virtual circuits), performance objectives that are referred to as the *traffic contract*.

The manner in which the ATM cells are multiplexed guarantees correct ordering of the upper-layer data and supports simultaneous transmission of multiple VCs in a single stream, as shown in Figure 3, page 6. (The single stream is passed to the SONET hardware as a single path and is explained in the following SONET section.) When the stream of cells arrives at its destination ATM layer, each conversation must be demultiplexed and reassembled before passing the data to the receiving application.

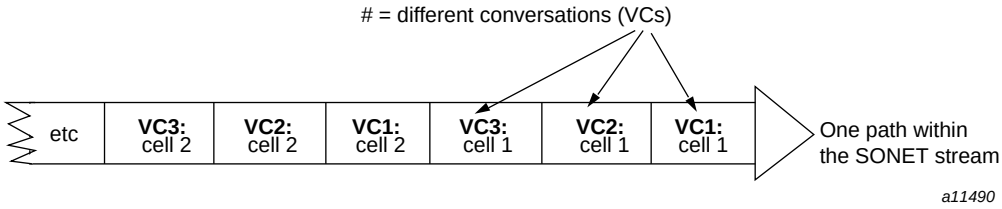
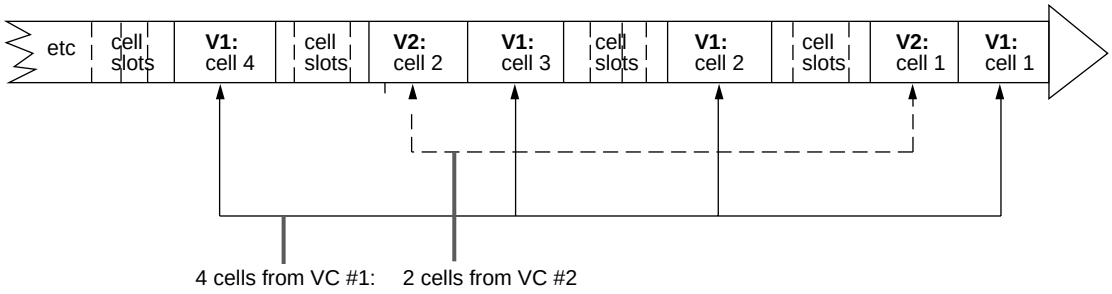


Figure 3. Multiplexed ATM Cells for Multiple VCs

Each VC within the single stream can be transmitted at a different rate. This is accomplished by taking cells from each VC's queue at a different rate. As the ATM hardware creates the single stream, it selects cells from the different VC streams in a manner that supports each channel's user-selected rate. For example, if the transmission rate for VC1 is twice the rate of VC2, the cells are selected and interleaved as shown in Figure 4, page 6, (instead of equally as shown in Figure 2, page 5, and Figure 3, page 6). In the example shown in Figure 4, page 6, when two cells have been transmitted for VC2, four cells have been transmitted for VC1.



Note: Transmission rate for VC #1 is 2 times the rate for VC #2.

Figure 4. Multiplexing Cells to Support Different Transmission Rates

The ATM Adaptation Layers (AAL) provide mapping between upper-layer formats and the ATM cell format. In addition, the AAL module handles the ATM cells in a manner that supports the selected class of service.[3] All AAL functionality occurs at the endpoints (not in the switches). Upper-layer

3 IRIS ATM currently supports only AAL5.

applications select a class of service (one of the AALs) from those summarized in Figure 5, page 7. AAL 5 is defined for high-speed data transfer and ATM signaling. AAL0 is an unofficial adaptation layer.

	AAL1	AAL2	AAL3/4	AAL5	AAL0
Bit rate	Constant	Constant or variable			
Timing	Source transmits clock; destination recovers clock from the bit stream.		No timing synchronization required.		
Data type	Connection‹ oriented			Connectionless	
Amount of ATM cell's payload used by AAL overhead	1 octet	still being studied	4 octets	none	none
Error detection	4‹ bit SNP	still being studied	10‹ bit CRC	32‹ bit CRC	none

a11492

Figure 5. AAL Service Classes: Types of Protocol Mappings for ATM

1.1.2 SONET

SONET defines the fiber-optic physical layer. It covers issues such as the specifications for the multimode fiber optic cable, loss characteristics on the connectors, clock recovery, the available formats for organizing data payloads, and the frame boundary delimitation. SONET provides a variety of data rates and supports numerous payload formats.

When discussing SONET rates, it is important to distinguish between the line or *signal rate* (that is, the rate on the fiber) and the rates of the various communication streams (referred to as *embedded transport rates*) being carried within the SONET stream. SONET supports signal rates that are multiples of the basic 51.84 Mbit/s synchronous transport signal (STS) rate, as summarized in Table 2, page 8. The embedded transport rates are always slower than (or equal to) the signal rate and include some of the more commonly used rates in the communications industry today: for example, 1.544 (DS1 and T1), 2.048 (CEPT), and 6.912 (DS2) Mbit/s.

Table 2. SONET Line Rates

Name	Also known as	Rate
OC1	STS-1, DS3, basic rate	51,840,000 bits per second (51.84 Mbit/s)
OC3[4]	STS-3	155,520,000 bits per second (155.52 Mbit/s)
OC12	STS-12	622,080,000 bits per second (622.08 Mbit/s)
OC48	STS-48	2,488,320,000 bits per second (2.48832 gigabits per second)

At the SONET level, the data stream logically consists of *n* separate paths (STS-1 streams), each carrying data of one type, encapsulated in STS-1 frames. The number of paths within a SONET stream is specified by the number in the SONET protocol's name. For example, SONET OC3 has three different paths (that is, three STS-1 streams) multiplexed within a signal rate of 155.52 Mbit/s. There is an exception to this. The concatenated formats of SONET (for example, OC3c and OC12c), have only one path and use an abbreviated form of the SONET frame. Within any SONET OC*n* stream, the multiple paths coexist through byte multiplexing; one byte from each path (each STS-1 frame) is transmitted, then another byte from each path is transmitted, as shown in Figure 6, page 9.

4 IRIS ATM supports only OC3c (155.52).

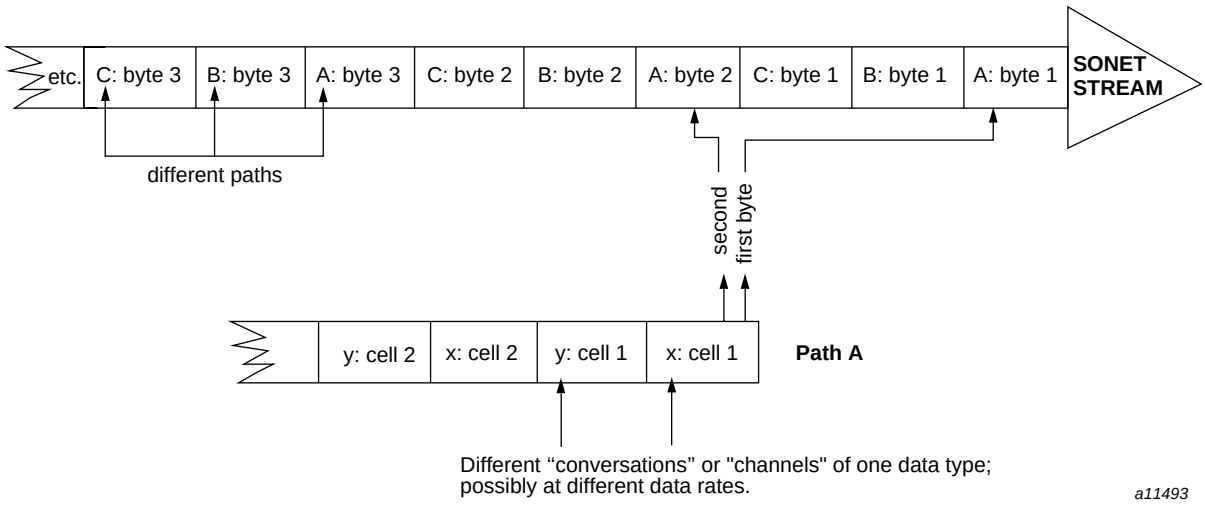


Figure 6. Byte Multiplexing within a SONET Stream

Note: The overhead associated with the SONET stream is not shown in Figure 6, page 9.

As explained in the preceding paragraphs, the basic physical building blocks for a SONET stream are bytes. Logically, however, the basic building blocks for a SONET data stream are *SONET frames* (also called STS-1 frames), as shown in Figure 7, page 10. Each STS-1 frame contains header and data for one path. The header contains protocol overhead data; the upper-layer data is carried in the *synchronous payload envelope (SPE)* portion of the frame. A SONET OC*n* stream uses larger frames constructed from *n* basic frames. For example, for SONET OC3 and OC3c, each frame carries three STS-1 SONET frames, as shown in Figure 8, page 10.

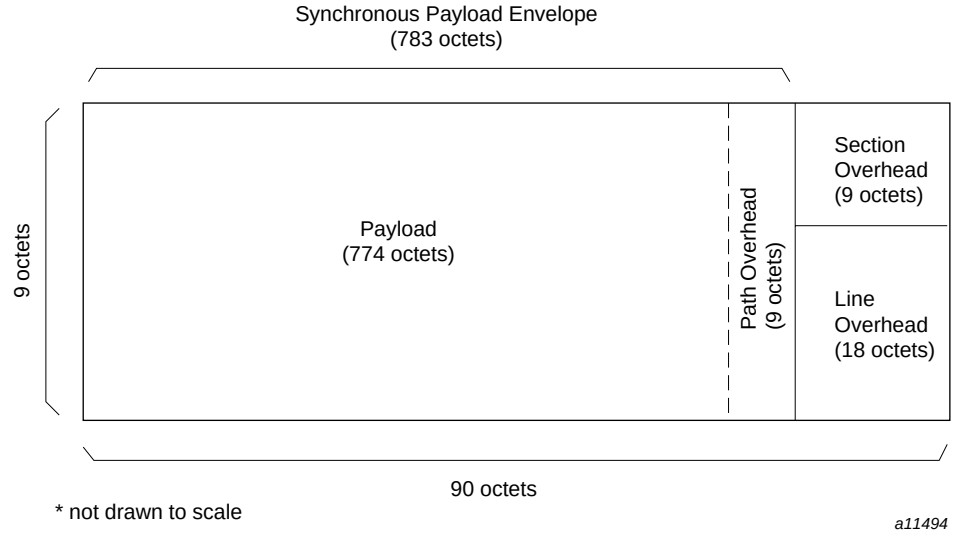


Figure 7. Basic SONET (STS-1) Frame

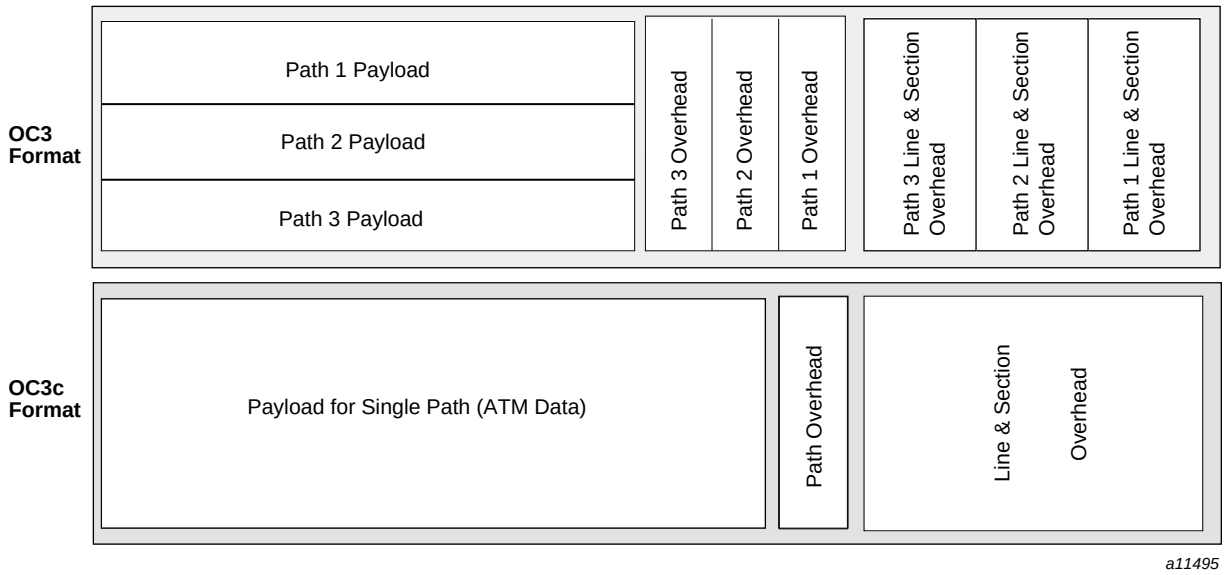


Figure 8. SONET OC3 Versus OC3c Frame Format

All of the data within any single SONET path must be of the same format. This is referred to as the *mapping* for the SPE. ATM is one of the available SPE mappings.[5] Since each path within the SONET stream is a separate logical entity, each path can be mapped differently from the other paths carried in that SONET stream.

Each path is capable of carrying a number of embedded streams at lower transport rates. One STS-1 SONET stream (for example, Path 1 shown in the OC3 frame of Figure 8, page 10) could carry twenty-eight 1.728 Mbit/s channels. When the SPE mapping is ATM, the separate paths are collapsed into a single concatenated path; for example, for the 155.52 Mbit/s rate, the SONET protocol is OC3c. In OC3c, both the line rate and the path rate are 155.52 Mbit/s. Figure 8, page 10, shows the difference between the triple-path format of OC3 and the collapsed, single-path format of OC3c.

1.2 IRIS ATM

The IRIS ATM product is a data communications interface controller board (hardware), and driver, protocol applications, and utilities (software) that provide data exchange through the Asynchronous Transfer Mode (ATM) protocol using ATM adaptation layer 5 (AAL5) for permanent virtual circuits (PVCs) and switched virtual circuits (SVCs) over a Synchronous Optical Network (SONET) physical layer. The product complies with the ATM Forum's *ATM User-Network Interface* standard, versions 3.0 and 3.1, including signaling and the interim local management interface (ILMI).

The product supports constant bit rate (CBR), variable bit rate (VBR), and best-effort traffic. The driver supports all standard IP applications through SVCs and PVCs using best-effort traffic contracts, in compliance with RFC 1577, *Classical IP Over ATM*. For environments that require CBR/VBR traffic (IP and non-IP), the IRIS ATM character device application programming interface (API) is provided so that customers can develop applications. The API is described in the *IRIS ATM API Programmer's Guide*. The product includes a VC management program (ATMARP) for IP-over-PVC configurations.[6]

The product provides ATM connectivity for the following platforms:

- CHALLENGE platforms
- Onyx platforms

5 The IRIS ATM board supports only ATM payloads.

6 The product does not include a VC management program for non-IP traffic. An application programming interface is provided so that customers who require non-IP traffic can develop applications.

- POWER CHALLENGE platforms
- POWER Onyx platforms
- Origin2000 platforms
- Onyx2 platforms

The IRIS ATM hardware must be installed by a Silicon Graphics system support engineer (SSE) or other person trained by Silicon Graphics. The *IRIS ATM–OC3c Board for Challenge or Onyx Installation Instructions* or the *IRIS ATM-OC3c 4Port XIO Board Installation Instructions* contains complete details for hardware installation.

The software installation and configuration described in this document can be done by customers or SSEs. This document, *IRIS ATM Configuration Guide* (shipped with the IRIS ATM software), provides software configuration details. The online documents, *IRIS ATM Release Notes* and *IRIX Admin: Software Installation and Licensing,* provide software installation instructions.

1.3 ATM Addresses

Two types of addresses are relevant for ATM networking: virtual path identifier/virtual channel identifier (VPI/VCI) addresses and network addresses. Permanent virtual circuits (PVCs) require only VPI/VCI addresses at the ATM layer. Switched virtual circuits (SVCs) require both types of addresses (although the VPI/VCI address is transparent to the user). The globally unique ATM network address is used by the signaling protocol to route the connection setup request from the calling party through one or more switches to the called party, but a local VPI/VCI is used (during the data transmission) by each switch along the route for demultiplexing and mapping between the virtual channel and the hardware resources that are allocated to the connection.

The following sections describe these two types of addresses.

1.3.1 VPI/VCI Addresses

The VPI/VCI address is a 3-octet value contained in the header of the ATM cell (as shown in Figure 9, page 13). This value identifies a virtual channel. The value is locally assigned by each transmitting station and is unique (and valid) for only one physical link of the virtual channel (for example, from the host to its switch or between two switches). The VPI/VCI value is replaced at each

switch along the virtual channel's span. This type of address is used for both PVCs and SVCs. For PVCs, it is the only ATM-level address required.

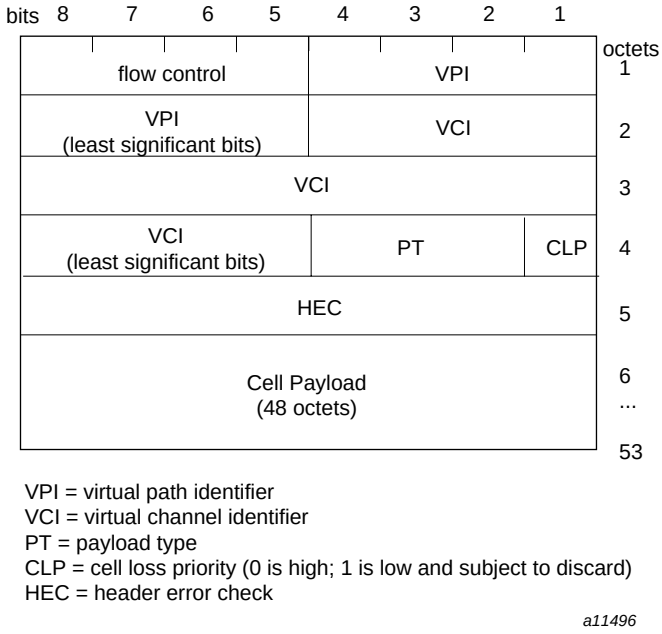


Figure 9. ATM Cell

1.3.2 Network Addresses

The ATM network address comes in two formats: a 20-octet value called ATM network service access point (NSAP) (shown in Figure 10, page 16) or an up-to-15-octet value called native E.164 (shown in Figure 11, page 17). The ATM network address is globally unique, meaning that it identifies one (and only one) endpoint within the entire world. This address, or a portion of it, is usually assigned to a port by its ATM switch. The NSAP format allows a system to support multiple endpoints using a single port by assigning local values to one portion of the address (as explained in more detail in the following paragraphs). The ATM network address is required for SVCs, but not for PVCs.

The ATM NSAP format (shown in Figure 10, page 16,) can carry any one of following three types of addresses:

- A country assigned address, which is indicated by the AFI field set to 39 and the IDI field containing a data country code (DCC).
- An E.164 address, which is indicated by the AFI field set to 45 and the IDI field containing a telephone-style E.164 number.
- An internally assigned address, which is indicated by the AFI field of 47 and the IDI filed containing an international code designator (ICD ATM).

The contents of the IDI field are represented in binary code decimal (BCD) notation. For example, the country code for the United States of America is 840 (decimal); this is represented in the IDI field by the binary sequence 1000 0100 0000. Each type of IDI value requires padding and all IDI field padding is done with four 1s. For example, the DCC code requires only 12 of the 16 bits in the IDI field and DCCs are padded on the right, resulting in a binary sequence of 1000 0100 0000 1111 for the United States of America.

Table 3, page 14, indicates the organization that assigns and defines the values for the three different IDI fields of ATM NSAP addresses.

Table 3. Values for the IDI Field of ATM NSAP Addresses

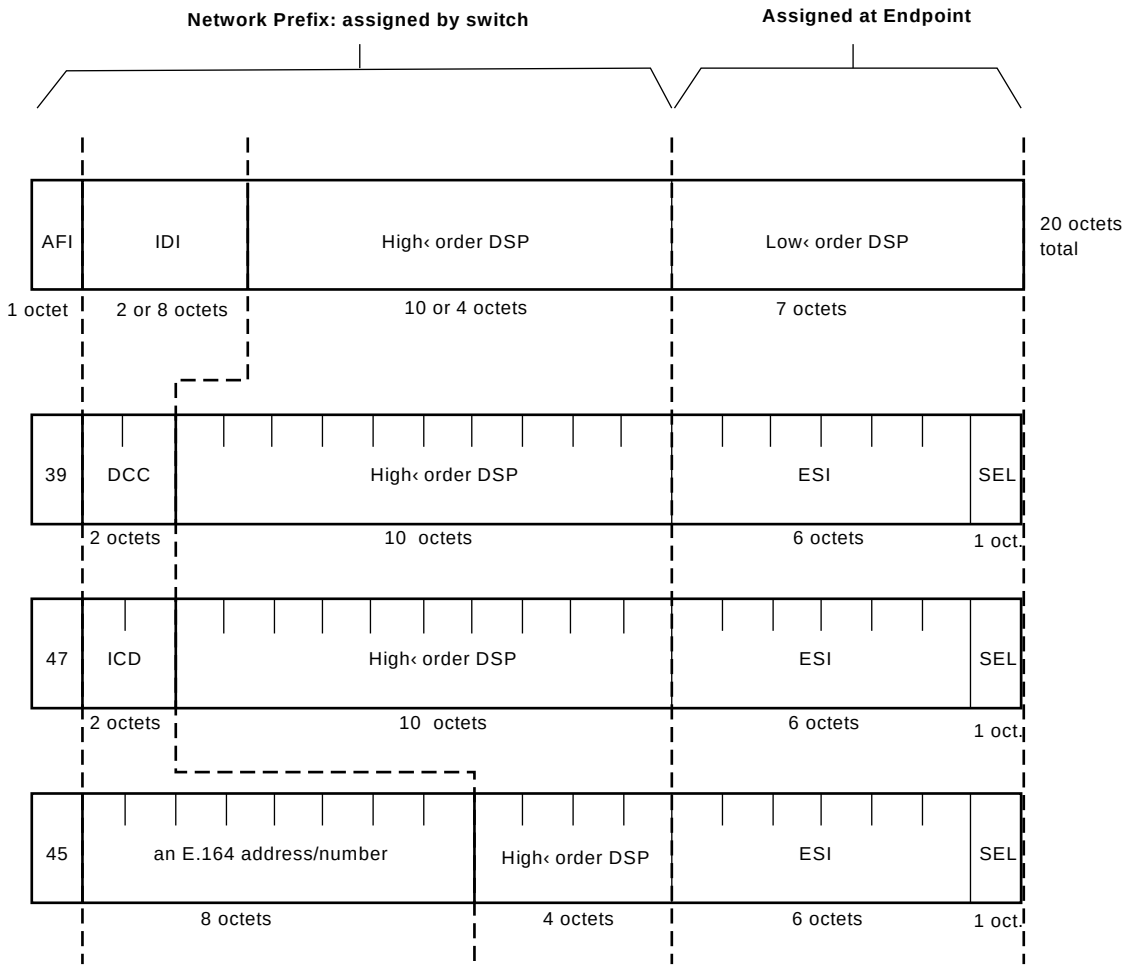
IDI field content	Standard
A data country code (DCC)	International Organization for Standardization: OSI specification ISO 3166
An E.164 number	International Telephone and Telegraph Consultative Committee: CCITT specifications I.330 and I.331
An international code designator (ICD)	British Standards Institute

The ATM NSAP address can be logically divided into two sections: that portion assigned by the switch and the portion assigned at the endpoint. The part assigned by the switch is referred to as the *network prefix*. It includes all fields of the address except the end system identifier (ESI) and end system selector (SEL) fields, as shown in Figure 10, page 16. The endpoint's interim local management interface (ILMI) module communicates with the adjacent switch to retrieve its assigned network prefix and to register its values for the ESI

field.[7] Switches ignore the SEL field; however endpoint software can assign values to this field to differentiate among multiple upper-layer endpoints.[8]

[7] For IRIS ATM, the ESI field is always a MAC address read from the IRIS ATM board.

[8] For IP-over-ATM, IRIS ATM sets the SEL field to a value that matches the logical IP network interface identification. For example, the ATM NSAP for logical network interface `atm0` uses SEL=0x00 but that for `atm4` uses SEL=0x04.



AFI = authority and format identifier (8 bits)
IDI = initial domain identifier (16 or 64 bits)
DSP = domain specific part (136 or 88 bits)

DCC = data country code (16 bits)
ICD = international code designator (16 bits)
ESI = end system identifier; can be a MAC address (48 bits)
IRIS ATM registers port's MAC address for this field.
SEL = end system selector; defined by local system, not by ATM standard (8 bits)
IRIS ATM software makes this field match the logical network interface number, so *atm1* uses SEL=0x01 and *atm47* uses SEL=0x2F.

a11498

Figure 10. ATM Address: The NSAP Format

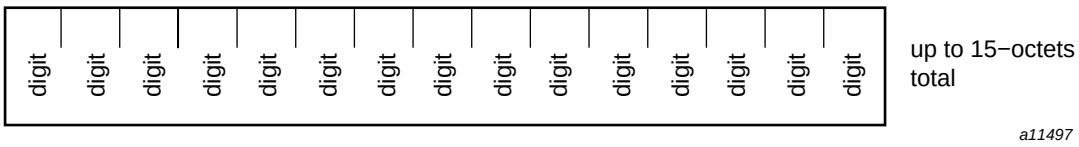


Figure 11. ATM Address: The E.164 Format

1.4 Virtual Channel Connections

ATM data is carried logically within a *virtual channel* (VC) and physically by the *virtual channel connection (VCC)* which is a sequence of physical links stretching from the source endpoint to the destination endpoint, passing through one or more ATM switches. The physical links that make up the virtual channel connection are not known to any one instance of the ATM layer; however, the properties of the full-length connection are important to an ATM network administrator. The following sections describe the different methods for setting up VCs and the parameters that describe VC and VCC functionality.

1.4.1 Traffic Contract

Each VC has a *traffic contract* associated with it. The traffic contract determines the performance characteristics of the data transmission. The two parameters that are always included in a traffic contract are the transmission rate (expressed in *ATM cells* per second) and the *quality of service (QoS)*. Other performance parameters can be included, for example, *cell delay variation (CDV)*, *cell transfer delay*, and *cell loss ratio*.

Since transmission rates are expressed in ATM cells per second, it is useful to know that one ATM cell carries 48 bytes of upper-layer data. For example, in order for an upper-layer to transmit (or receive) 3.5 megabits of data per second, the traffic contract must specify about 9115 cells per second (9115 cells * 48 bytes in each cell * 8 bits in each byte = 3,500,160 bits). If the upper-layer data includes an encapsulated protocol, some of the 48 bytes may contain non-ATM overhead, like TCP/IP headers.

The QoS classes are as follows:

- Class 1 for constant bit rate traffic (CBR), like video and audio
- Class 2 for variable bit rate (VBR), like compressed video and audio

- Class 3 for connection-oriented data, like Frame Relay
- Class 4 for connectionless data, like IP network traffic

The manner in which the traffic contract is negotiated depends on whether the endpoints are using a permanent virtual circuit (PVC) or a switched virtual circuit (SVC), as explained in Section 1.4.2 and Section 1.4.3, page 20.

1.4.2 Permanent Virtual Circuits

A *permanent virtual circuit (PVC)* is a long-term (permanent) communication channel between two ATM endpoints. The channel can directly connect two ATM endpoints or can involve any number of intermediate switches between the two endpoints. PVCs are created during a relatively difficult installation and setup procedure. The traffic contract is negotiated, person-to-person or as-advertised, but in all cases, before the installation and setup takes place. The traffic contract's performance parameters are either built into the equipment or are configured, manually, by a network administrator. Each node in the network must be configured to conform to the negotiated traffic contract and the contract cannot easily be changed. The sequence of physical links for each PVC must be planned and manually configured. First, the existence of a complete physical connection must be verified. Then, at each port along the route, an address must be created to identify the resources being reserved for this PVC; this address usually consists of a VPI/VCI value and a port identification. Before a VPI/VCI value is selected for a link, the administrator must verify that the value is available at both ends of the link. For the example PVC shown in Figure 12, page 19, four address mapping tables (one at each node) must be configured manually with the following bidirectional mappings:

- At Endpoint A: upper-layer network address and resource address for link 1
- At Switch1: resource address for link 1 and resource address for link 2
- At Switch2: resource address for link 2 and resource address for link 3
- At Endpoint B: resource address for link 3 and upper-layer network address

PVCs require significantly less software complexity and overhead than SVCs; however, they require significantly more administrative time for planning and configuring. PVCs are appropriate for environments in which the traffic contract and the point-to-point connections (PVCs) are well-defined and will remain stable for a reasonably long period of time. PVCs are required if any switch or host along the path does not support SVCs.

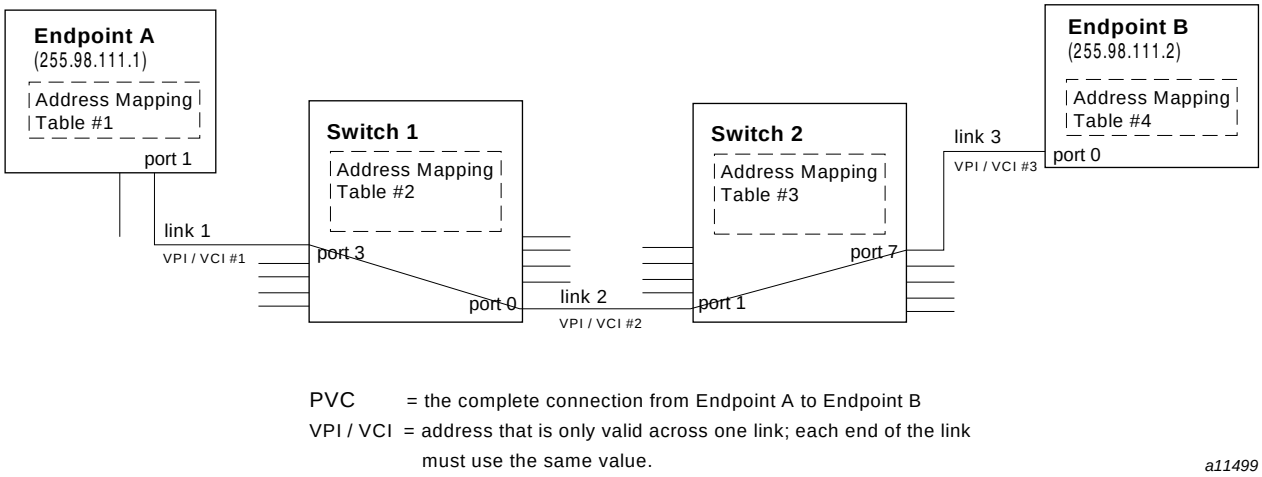


Figure 12. General Overview of PVC Configuration Tasks

As an example of the manual configuration entries required for every PVC, Figure 13, page 19, shows sample address mapping tables for the single PVC shown in Figure 12, page 19.

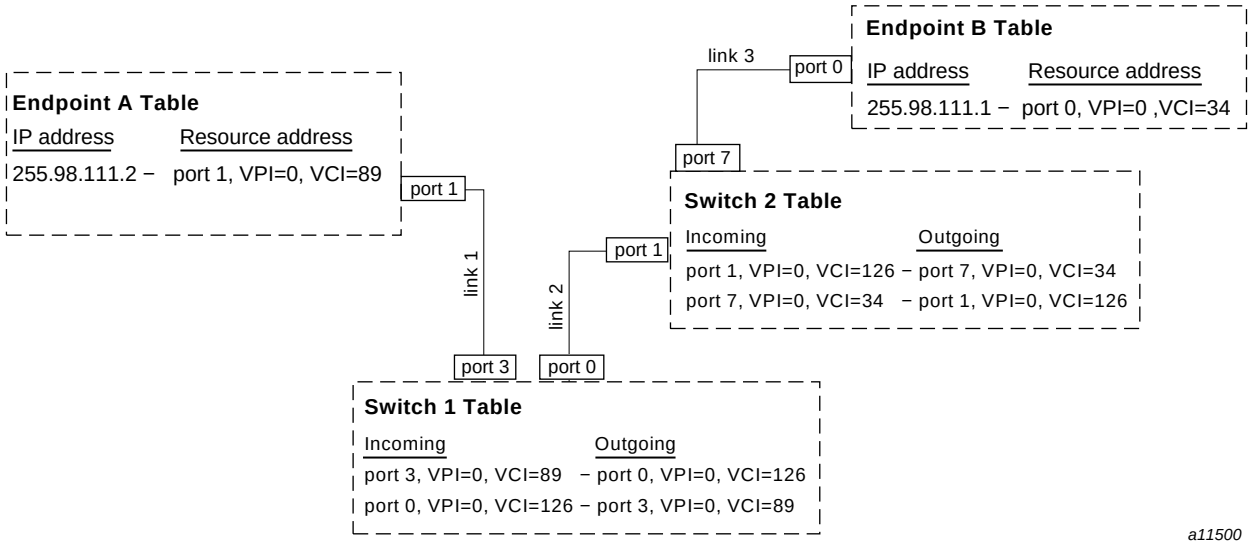


Figure 13. Example of Address Mapping Tables for One PVC

1.4.3 Switched Virtual Circuits

A *switched virtual circuit (SVC)* is created and torn down dynamically (more or less in real time), as requested by an upper-layer application. The switches, the ATM signaling software, and, for IP traffic, the ATMARP software, automatically handle most of the negotiable parameters and operation management, including the following operations:

- ATM address registration
- Address resolution
- Discovering a route between the two endpoints
- VPI/VCI assignment at each link
- Resource allocation along the entire route
- Negotiation of the traffic parameters

Due to this automation, an SVC environment requires much less administrative time than a PVC configuration.

The two endpoints for an SVC are known as the calling party and the called party. The calling party is the endpoint that originates the setup request for the SVC. Each SVC is bidirectional and is, in fact, two *virtual channels* (VCs)—a forward VC and backward (or return) VC. The forward VC carries data from the calling endpoint to the called endpoint; the calling endpoint transmits on this channel, the called endpoint receives on this channel. The backward VC carries data in the opposite direction, so the calling endpoint receives on this channel while the called endpoint transmits on it. Three topics related to SVC operation are discussed in more detail in the sections that follow:

- ATM signaling
- ATM UNI
- ATM ILMI

1.4.3.1 ATM Signaling

ATM signaling is the protocol that sets up and tears down an SVC. This protocol's full name is ATM user-to-network interface (UNI) signaling. A

calling endpoint uses ATM signaling to request a channel and to specify the *traffic contract* for the SVC. The setup request is sent to an adjacent node, which can be either an adjacent ATM switch (either public or private) or the called endpoint.[9] If the recipient of the setup request is an ATM switch (the network side of the UNI interface), it uses one of the following switch-to-switch protocols to discover a route to the called party and to forward the message through the network to the destination: (1) the network node interface (NNI) protocol with dynamically maintained route information, or (2) the interim interswitch signaling protocol (IISP) with manually configured route lookup tables. (Usage of UNI and NNI protocols between different types of nodes is shown in Figure 14, page 22.) For some parameters of the traffic contract, the called endpoint can modify the contract by selecting among a list of possible values before the connection is completely set up. The traffic contract can be different for the forward and back channels of an SVC.

9 The term *public* describes services that are offered to the general public by equipment supplied by a public service company (such as a telephone company). *Private* indicates services that are offered by a private entity to a restricted set of users (for example, to employees of a company).

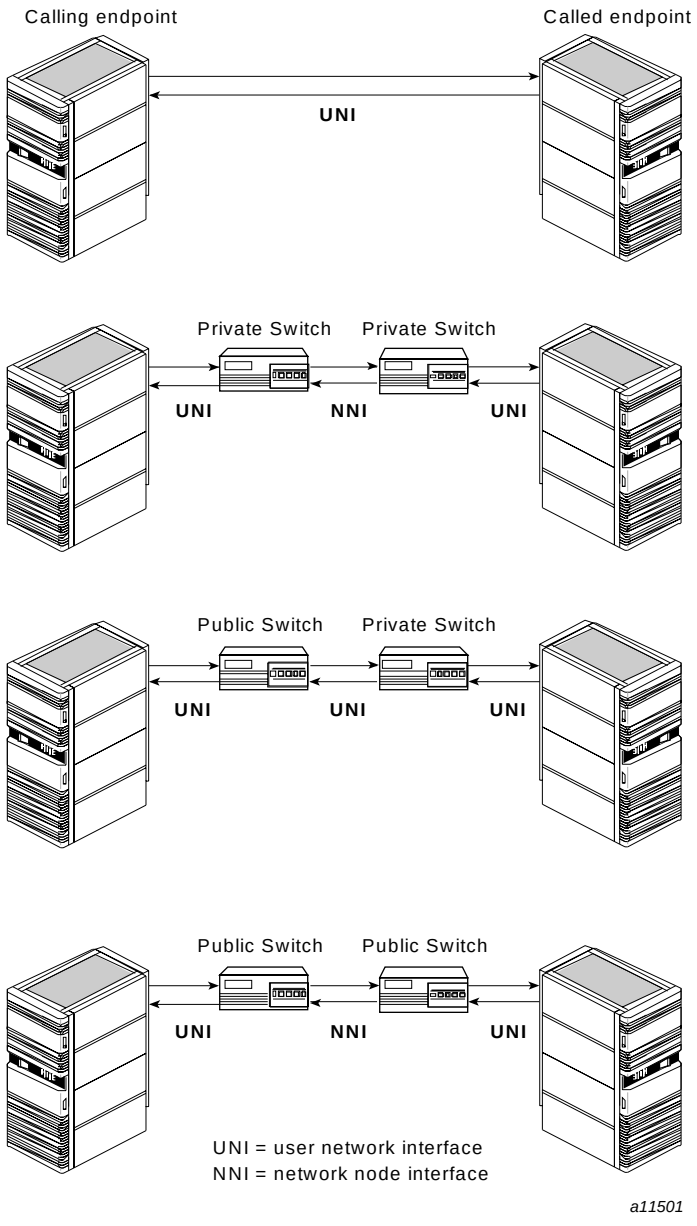


Figure 14. Use of UNI and NNI Protocols for ATM Signaling

The basic signaling messages used for managing SVCs are described in Table 4, page 23.

Table 4. Mandatory ATM UNI Signaling Messages

ATM signaling message	Who originates message	Who receives and processes message
SETUP: Requests that a bidirectional SVC be created and specifies the traffic contract. Some contract parameters allow the called party to select among a list.	Any node implementing a UNI. Sent on forward channel.	The adjacent node (switch or called endpoint). When the recipient is a switch, the request is forwarded to the next hop on the route to the called party, and can be converted into the NNI format. The final switch gives the SETUP message to the called party. Each recipient of this message allocates resources and sets up the SVC, if it can.
CALL PROCEEDING: Indicates that the SETUP message was received. This message is optional.	Each node that receives a SETUP message. Sent on back channel.	The adjacent node (switch or endpoint).
CONNECT: Indicates that the SVC has been set up completely between the two endpoints. This message contains any traffic parameters that were negotiated.	The called party after receiving a SETUP message and after setting up the SVC. Sent on back channel.	The adjacent node (switch or calling endpoint). When the recipient is a switch, the message is forwarded to the next hop along the route going back to the calling party and can be converted into the NNI format. The final switch gives the CONNECT message to the calling party.
CONNECT ACKNOWLEDGE: Indicates that the SVC is set up and functional in both directions.	The calling party after receiving the CONNECT message. Sent on forward channel.	The adjacent node (switch or called endpoint). When the recipient is a switch, the message is forwarded to the next hop on the route to the called party, and may be converted into the NNI format. The final switch gives the message to the called party.
RELEASE: Requests that the SVC be torn down.	Either endpoint of an SVC. For a calling party, sent on forward channel. For a called party, sent on back channel.	The adjacent node (switch or endpoint). When the recipient is a switch, the message is forwarded to the next hop on the route to the other endpoint, and can be converted into the NNI format. Upon receipt of this message, each node tears down its local resources for this SVC.

ATM signaling message	Who originates message	Who receives and processes message
RELEASE COMPLETE: Indicates that the SVC has been torn down. As soon as this message is transmitted, all references to the SVC are erased.	Each node that received a RELEASE message. Sent on the opposite channel from the RELEASE message.	The adjacent node (switch or endpoint).
STATUS ENQUIRY: Requests status information about the SVC.	An endpoint or its adjacent switch. Sent on either channel.	The adjacent node (switch or endpoint).
STATUS: Indicates the status of the SVC at this node.	An endpoint or switch that received a STATUS ENQUIRY message. Sent on the opposite channel from the STATUS ENQUIRY message.	The node that generated the STATUS ENQUIRY message.

Four messages are involved in creating an SVC: SETUP, CALL PROCEEDING, CONNECT, and CONNECT ACKNOWLEDGEMENT. Figure 15, page 25, shows the order in which these messages are processed by the different nodes.

SETUP is the first message in the creation of an SVC. This message can be originated by either the source or the destination endpoint of a data transaction. As each switch along the path between the two endpoints receives the SETUP request, it may respond with a CALL PROCEEDING acknowledgment message, as shown in Figure 15, page 25. Each switch sets up its links for the SVC connection by allocating resources for a bidirectional connection at the specified traffic contracts. Once the resources are allocated, the switch forwards the SETUP request to the next node enroute to the other endpoint. When the SETUP request has been received and successfully processed by the called endpoint, the endpoint sends a CONNECT message, which is propagated along the return (backward) VC to the endpoint that originated the SETUP request. If a switch cannot set up the requested SVC, it does not forward the SETUP message and instead follows its CALL PROCEEDING message with a RELEASE message that causes all the links for that SVC to be torn down. If the called endpoint cannot match the requested traffic contract or does not want to accept the connection, it sends a RELEASE instead of a CONNECT message. Figure 16, page 26, shows the bidirectional SVC (two VCs) that is the result of a successful SETUP request.

Note: The term *forward* is always used to describe the channel that carries data from the calling party to the called party. The term *backward* always refers to the channel that carries data from the called party to the calling party.

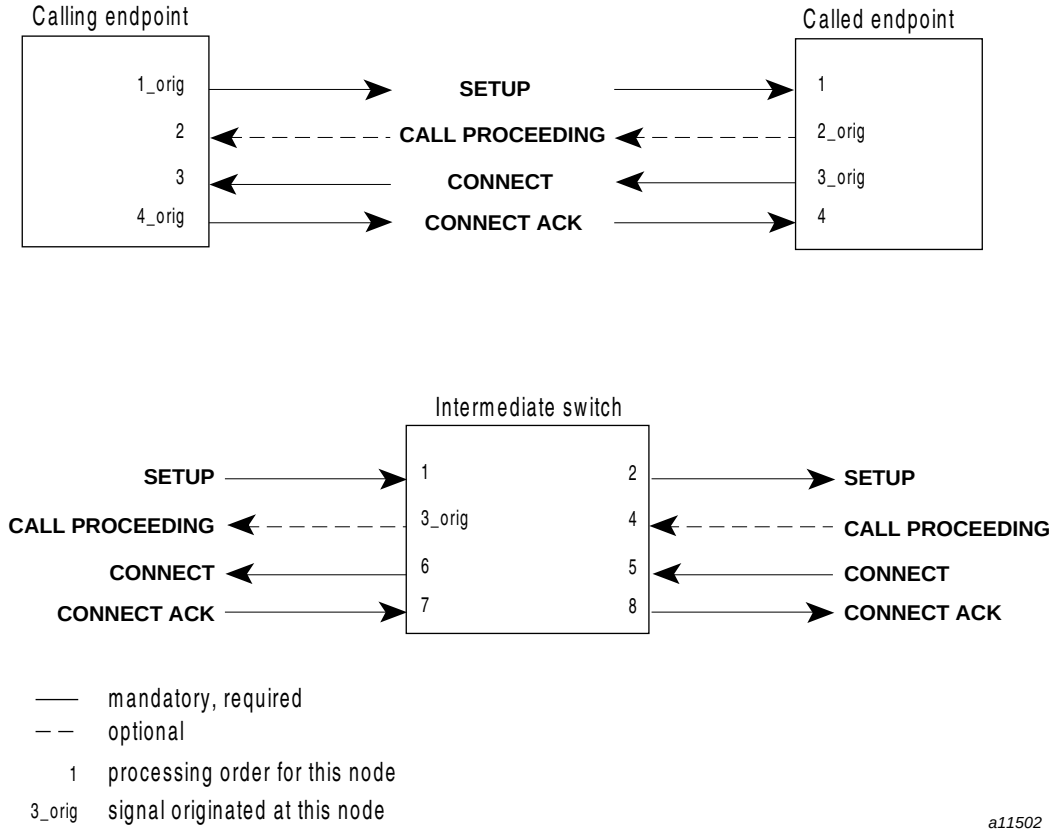


Figure 15. ATM Signaling Messages for Creating an SVC

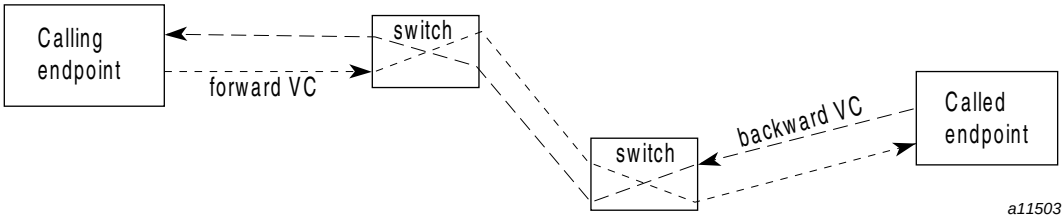
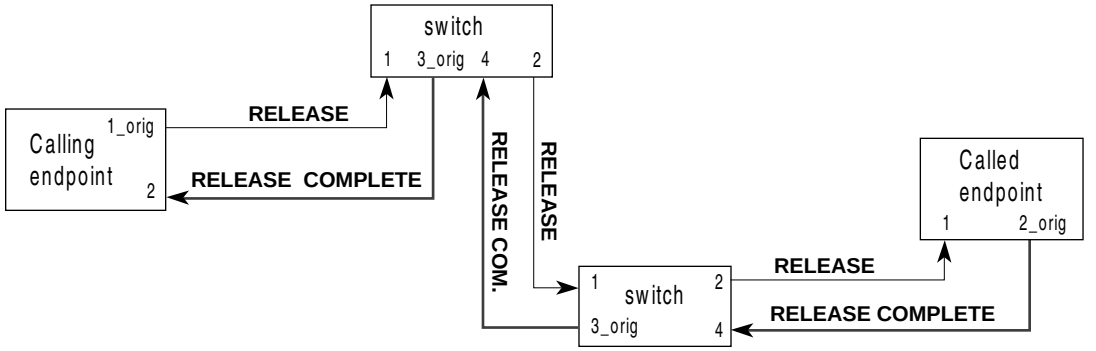


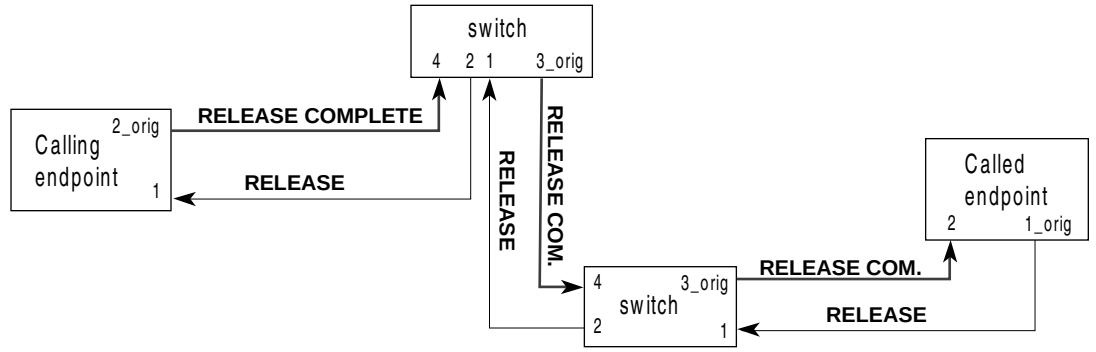
Figure 16. Result of a Successful SETUP Request for an SVC

When either endpoint wishes to terminate the connection, it generates a RELEASE message that causes each node along the connection to tear down the two VCs and pass the message on to the next node. As each node completes its tear down and frees its resources, it sends a RELEASE COMPLETE message to the adjacent node from which the RELEASE message came. Figure 17, page 27, shows the use of these messages for two cases: (A) the case in which the calling endpoint initiates the release, and (B) the case in which the called endpoint initiates the release.

A. SVC release initiated by calling party



B. SVC release initiated by called party



= order in which this node processes signals

#_orig = signal originated at this node

a11504

Figure 17. ATM Signaling Messages for Tearing down an SVC

1.4.3.2 ATM UNI

For IRIS ATM, the *ATM user-network interface (UNI)* refers to the complete set of functionality that controls how an ATM endpoint (the *user*) interfaces with a switch (the *network*). Each ATM port requires one ATM UNI. The IRIS ATM signaling software (`atmsigd`) consists of a number of modules that run as an

IRIX daemon and handle the various functions and protocols related to the UNIs on a system.

For every physical ATM port, the ATM standard requires one instance of an ATM UNI. For IRIS ATM, `atmsigd` performs this function. For each UNI, `atmsigd` creates a software stack made of the four following modules (shown in Figure 18, page 29). Each UNI has its own set of the lower three modules; all the UNIs within a system share a single instance of the overall control module.

- Overall control
- Signaling (Q.2931)
- Service specific convergence protocol (SSCOP, also known as QSAAL)
- ATM adaptation layer 5 (AAL5)

For each UNI, `atmsigd` creates a PVC to the switch using the following default address: VPI=0 and VCI=5. The signaling module, as shown in Figure 18, page 29, uses this PVC for its own communications with the switch (for example, setup and teardown requests for SVCs). The VPI/VCI address of this PVC is configurable. Overall, the traffic on this PVC is sporadic and takes up only a small portion of any port's bandwidth; however, the higher the PVC's configured rate, the larger the percentage of the port's total bandwidth that can be occupied by this overhead at a specific point in time (that is, when there is UNI signaling occurring).

ATM network administration and management is based on the existence of an ATM management information base (MIB) that is managed by ILMI software via a PVC to the switch. For each UNI, there is one MIB and one ILMI PVC.

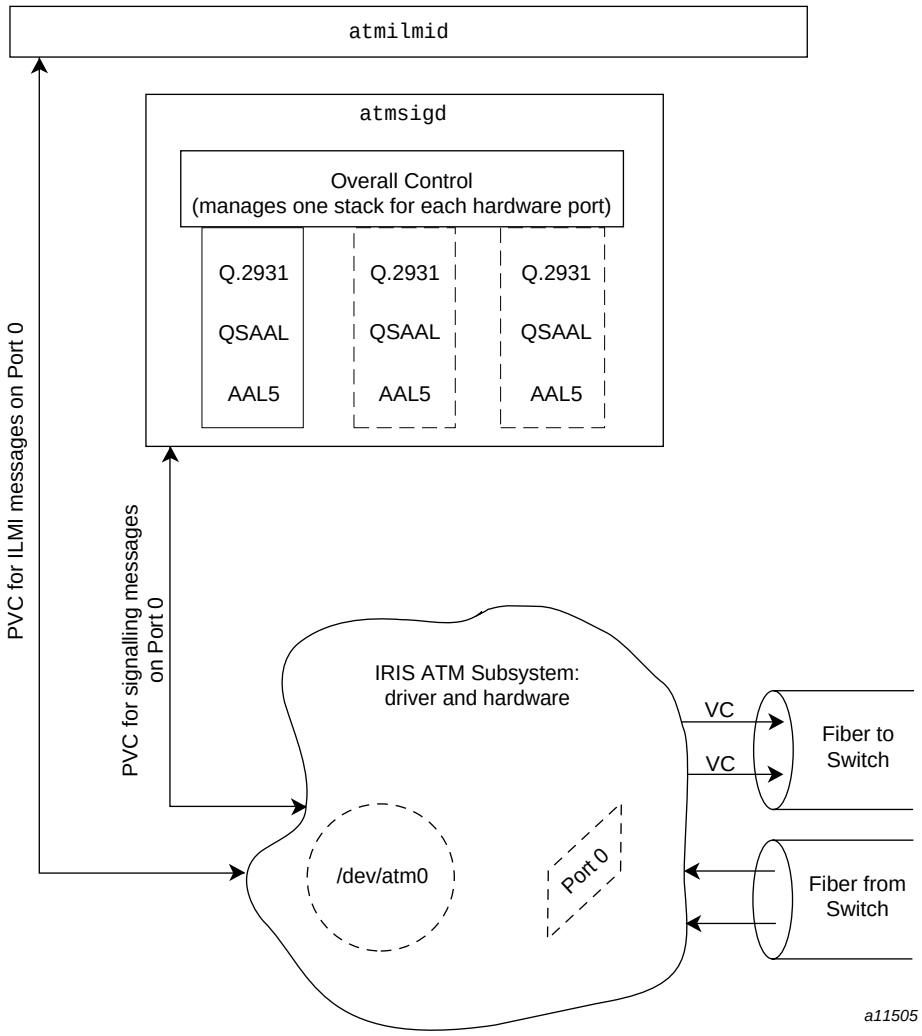


Figure 18. The IRIS ATM Signaling and ILMI Daemons

1.4.3.3 ATM ILMI

The ATM standard specifies an interim local management interface (ILMI) to handle address registration and assignment, and status reporting (shown in Figure 18, page 29, and Figure 19, page 31). To exchange status information, ILMI implementations use the simple network management protocol (SNMP, RFC 1157). ILMI implementations store some of the information they collect in

management information databases (MIBs) that users can view. There is one MIB for each UNI (that is, each physical connection). The ATM MIBs contain objects and tables that are specified by the *ATM User-Network Interface Specification* standard. The module in IRIS ATM that performs these duties is `atmilmid`. Like `snmpd`, `atmilmid` is an administrative process (IRIX daemon) that acts as an SNMP agent, managing MIBs and exchanging information with other ILMI agents. The `atmilmid` also functions as a subagent to the main SNMP agent (`snmpd`) so that the ATM MIBs can be viewed with standard SNMP MIB browsers.

The `atmilmid` daemon responds to requests from other ILMI modules (for example, those that reside on adjacent switches) as well as requests from the local main SNMP agent, as shown in Figure 19, page 31. To communicate with the local SNMP agent, `atmilmid` listens on a user datagram protocol (UDP) socket. To communicate with each adjacent ILMI agent, the `atmilmid` uses a permanent virtual circuit (PVC) on each port with the following default address: VPI=0 and VCI=16. (The VPI/VCI address for the PVC and the socket address are both configurable.) Overall, the traffic on this PVC is sporadic and takes up only a small portion of any port's bandwidth; however, the higher the PVC's configured rate, the larger the percentage of the port's total bandwidth that can be occupied by this overhead at a specific point in time (that is, when there is ILMI overhead communication occurring).

During startup, `atmilmid` opens a PVC to each adjacent switch and contacts each ILMI agent, as shown in Figure 19, page 31. From each ILMI agent, `atmilmid` obtains the switch-assigned portion for its ATM address and it registers its locally assigned portion (MAC address). If the ILMI agent on the switch is not available, `atmilmid` completes this task as soon as that agent comes online. If the request times out, `atmilmid` looks for a locally configured ATM address. If no ATM address is available, `atmilmid` uses a null address.

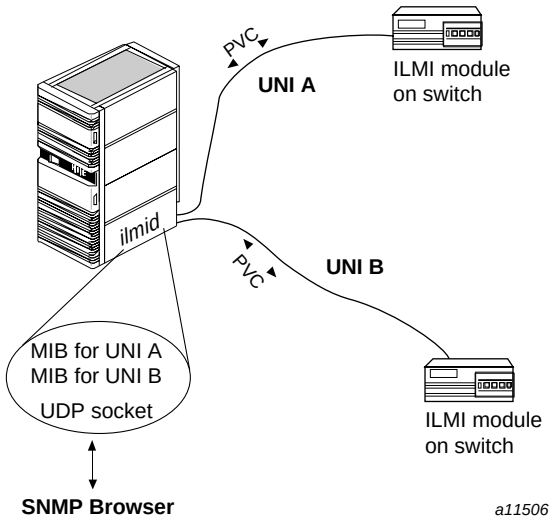


Figure 19. Relationship of ILMI Module to UNIs

After this initialization procedure, the `atmilmid` daemon uses its PVC during normal operation to exchange status information with adjacent ILMI agents. The `atmilmid` daemon maintains (in memory) one MIB for each UNI. Each UNI MIB contains the adjacent switch's table of supported network prefixes and UNI status objects, as specified by the ATM Forum standards. You can view the ATM MIBs by using any application developed for viewing SNMP MIBs (for example, the IRIXpro Browser\xb9).

1.5 ATM Standards

The documents that specify the ATM standards to which IRIS ATM complies are as follows:

- *ATM User-Network Interface Specification, Version 3.0*, released by The ATM Forum Technical Committee, September 1993.
- *ATM User-Network Interface Specification, Version 3.1*, released by The ATM Forum Technical Committee, September 1994.

1.6 IP and ATM in IRIS ATM

IP and ATM are both protocols that include network layer processing that supports routing. In ATM environments of the future, ATM will function as the main network layer and IP will be overlayered, as shown in Figure 1, page 3. In these future, large, globally connected environments, routing will be done by ATM. In the meantime, while ATM routing is not fully standardized and implemented, and ATM networks are not globally connected, the current standards for IP routing and address resolution can be used. The method for implementing this "classical" functionality is defined by RFC 1577.[10] IRIS ATM logical network interfaces conform to RFC 1577 rules and guidelines, whether using SVCs or PVCs, as long as the onsite configuration is set up according to the RFC 1577 guidelines. If a site wants to create a nonconforming configuration, IRIS ATM also supports this, as explained in more detail in Section 1.6.3, page 40.

RFC 1577 specifies a set of rules for implementing IP routing and address resolution over ATM. Implementations that are compliant with RFC 1577 are termed *classical IP* because the design treats IP networks in ATM environments as if they were still local area networks (that is, as if the systems sharing a subnetwork address were a collection of systems physically connected to a shared communication medium).[11] The design specified by RFC 1577 differs slightly for permanent virtual circuits (PVCs) and switched virtual circuits (SVCs), and for VCs that use or do not use LLC/SNAP encapsulation, as discussed in the following paragraphs.

One important difference between configuring IP in legacy LAN environments as compared to classical IP-over-ATM environments, is that each ATM physical port can support multiple subnetworks, whereas in legacy LANs each physical connection supports one logical network interface. In IP implementations over shared, broadcast mediums, such as Ethernet or FDDI, the grouping of stations into a subnetwork is a physical event (connection to a LAN). In IP-over-ATM, the grouping of stations into a subnetwork is not a physical event, but an administrative one (simple assignment of addresses). This difference is due to the fact that ATM allows the total bandwidth of any single connection (port) to be subdivided and separately addressed into simultaneously functioning virtual channels; each channel can carry data to and from a different subnetwork.

10 There are a number of proposals for designs and protocols that handle routing, address resolution services, and other network services over ATM. RFC 1577 is the one that defines the standard for "classical IP."

11 Two systems share the same subnetwork address when the network portions of their IP addresses match and they are using the same subnet mask.

1.6.1 IP over SVCs

In IP networks, it is common to think of each host (that is, each system on a network) as identified by two unique addresses: a logical IP address and a hardware MAC (or Ethernet) address. This view is, however, inaccurate. It is more accurate to think of the MAC hardware address as identifying only the physical connection to the network and the IP address as identifying one upper-layer software endpoint (for example, a logical IP network interface, such as `et0` or `atm0`). When described this way, it is conceivable to have multiple upper-layer endpoints in a system, all sharing the same physical connection. And this is exactly what ATM makes possible: numerous IP interfaces all sharing one ATM port. The following paragraphs describe exactly how this is done.

In ATM switched virtual circuit environments, each upper-layer endpoint is identified by an ATM address, as described in Section 1.3, page 12. In concept, an ATM address is a network layer address, not a hardware address. However, in many ATM implementations (including IRIS ATM), an ATM address is partially a hardware address because it includes the hardware address (MAC address) of the ATM port. All of the fields of the ATM NSAP except the end system selector (`SEL`) field constitute the ATM port's address; the entire ATM NSAP, including the `SEL` field, is the endpoint's address. The `SEL` field is used to distinguish between the upper-layer endpoints that share that port.

To do IP-over-SVCs, it is necessary to map between IP and ATM addresses. RFC 1577 specifies a method for doing this. The method uses the ATM address in place of the standard Ethernet or MAC hardware address, and describes a protocol, ATMARP, for registering and discovering the mappings, and for maintaining the IP-to-ATM address resolution table. The RFC 1577 design is shown in Figure 20, page 34. The ATMARP protocol uses standard IP address resolution protocol (ARP) (RFC 826) to discover ATM addresses when only the IP address is known, and inverse IP ARP (RFC 1293) to discover IP addresses when only the ATM address is known.

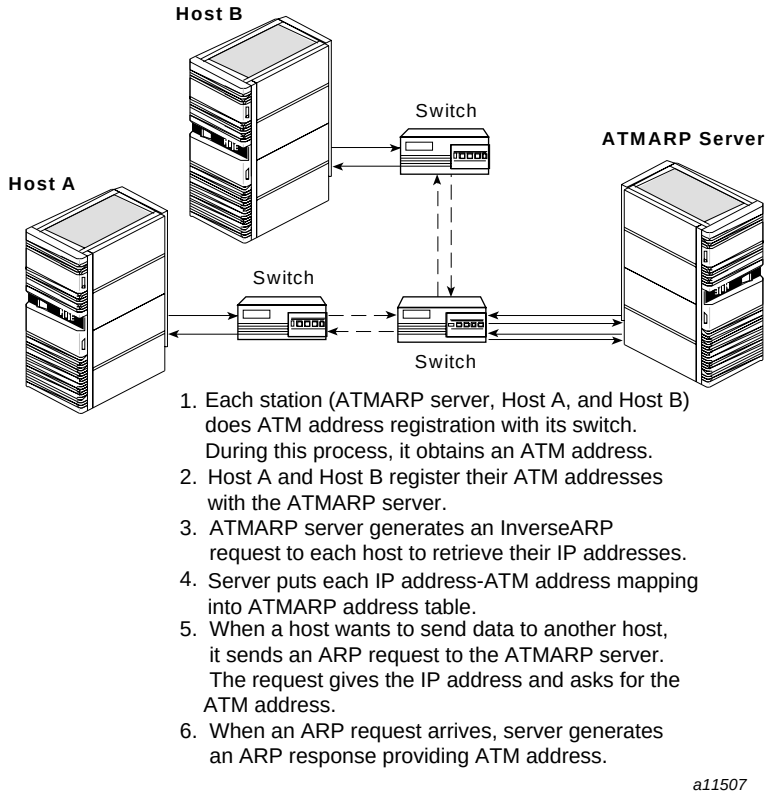


Figure 20. ATM Address Resolution Events

In the RFC 1577 design, all IP hosts that use the same network address and subnet mask (that is, the same subnet address) are considered members of a single *logical IP subnetwork (LIS)*. Each LIS can be thought of as logically similar to an Ethernet local area network, even though the members of the LIS do not have any special physical relationship to each other and can even be separated by multiple ATM switches. Each LIS member must be known to other members of that LIS by a single ATM address. For IRIS ATM, this rule means that all traffic between an IRIS ATM endpoint and other members of a specific LIS must travel over the same physical port, as shown in Figure 21, page 35. To contact an IP host that is not a member of the same LIS, a router must be used, even when it is physically possible for the IP host to be contacted directly through the ATM switch, as illustrated by network 255.100.8, which is shown in Figure 21, page 35.

Within each LIS, one (and only one) host acts as the address resolution server and maintains the IP-to-ATM address resolution table. Upon request, this server provides the ATM address for any other member of that LIS. Each LIS member (IP endpoint) registers its IP address and ATM address with the ATMARP server. When the ATM address is in the ATM NSAP format, a different ATM address can be registered for each of the local IP addresses.[12] This is accomplished by using the SEL field (which is not interpreted by ATM switches) to distinguish among the IP upper-layer endpoints. For example, the ATM address for atm2 using port 0 might consist of the ATM address for port 0 (network prefix and MAC address) plus the 8-bit SEL field set to 00000010 binary (which is 2 in decimal format).

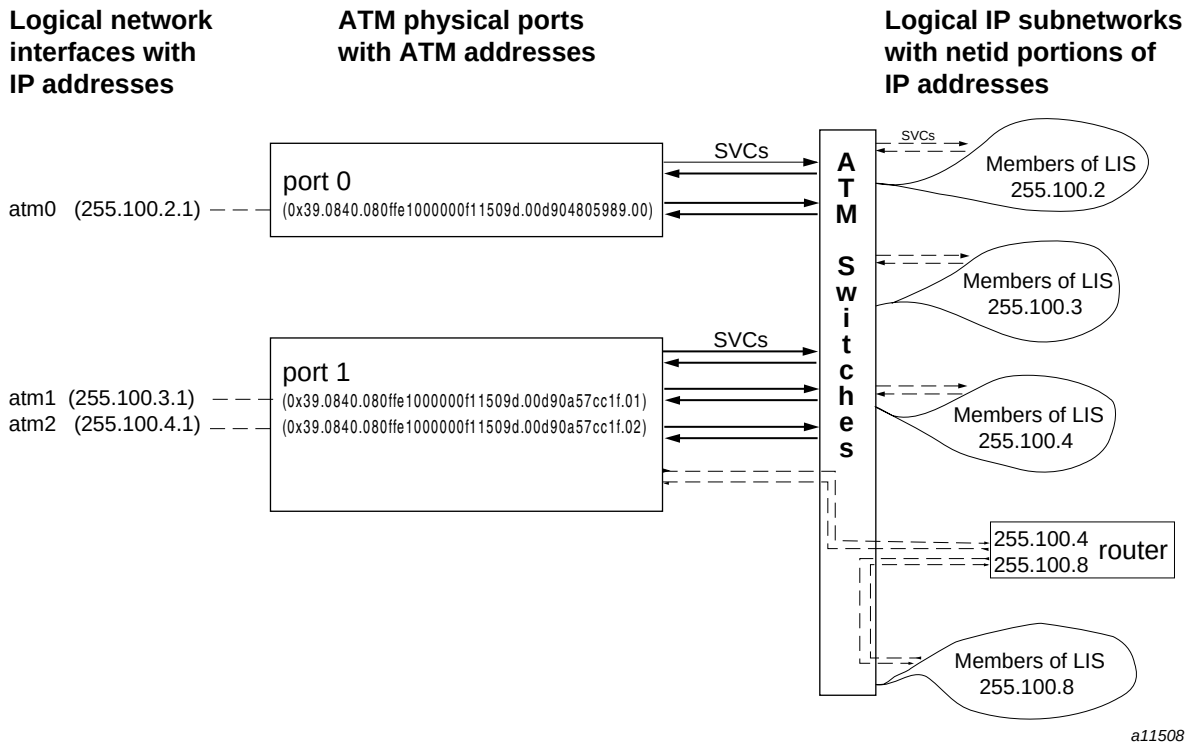


Figure 21. Multiple IP Interfaces Using Each ATM Physical Port with SVCs

12 IRIS ATM does this automatically by setting the SEL field set to the same value as the logical network interface number. For example, for atm0, SEL is 0; for atm2, SEL is 2.

Because the logical network interfaces are upper–layer, nonphysical entities and the ATM port can simultaneously handle multiple communication channels, a single physical ATM port can service more than one logical network interface, as shown in Figure 21, page 35. (Each logical network interface identifies one member of an LIS.) In this figure, port 1 (board unit 1) services both `atm1` (a member of LIS 255.100.3) and `atm2` (a member of LIS 255.100.4). IRIS ATM can simultaneously support up to 64 logical network interfaces. These can all share a single physical port, or they can be distributed among a number of ports. During the SVC configuration process, the network administrator assigns each logical network interface to one (and only one) port. Figure 21, page 35, illustrates a system with three logical network interfaces (each one is a member of a different LIS) and two ATM physical ports.

The IRIS ATM subsystem handles IP-over-SVC address resolution with the following mechanisms:

- The IRIS ATM ILMI (`atmilmid`) module, at startup, automatically communicates with the adjacent switch on each ATM hardware connection (port) to create an ATM address for itself. On each ATM port that is using an ATM NSAP address, the `atmilmid` obtains the network-prefix portion of the ATM NSAP from the switch and registers its local portion (a MAC address). If the address request on any port times out without having received an address from the adjacent system, IRIS ATM uses the address configured in the `/var/atm/atmilmid.conf` file.

- At startup, the IRIS ATM initialization script (`init.d.atm`) looks in the `/var/atm/ifatm.conf` file for the address resolution server and the port assignment for each logical network interface. (Each logical network interface is a local endpoint for one LIS.) The software then opens up an SVC to each ATMARP server and registers its IP-to-ATM address mapping (using ATMARP and LLC/SNAP encapsulation). This SVC is kept open for ATMARP communications. The software reopens the SVC if it goes away at any time.

  When a port's ATM address is in the ATM NSAP format, a unique ATM address is mapped to each of the local IP addresses that use that port. The registered ATM address consists of the port's ATM address (network prefix and MAC address) plus the `SEL` field set to the logical network interface number. For example, the ATM address for `atm2` using port 0 consists of the port's network prefix and MAC address plus the 8-bit `SEL` field set to 00000010 (binary).

- For each IP-over-SVC transmission request, the IRIS ATM software looks first in its local cache of IP-to-ATM address mappings. If the address is not

there, the software uses its SVC to the address resolution (ATMARP) server to discover the ATM address that corresponds to the IP address. Once the ATM address is known, a bidirectional SVC is created to the endpoint and data can be exchanged. If the SVC becomes idle (that is, has not carried any data for a configurable timeout period), it is torn down.

- If the IRIS ATM software has been configured to function as an ATMARP server, the software maintains the IP-to-ATM address resolution table, and responds to client requests for ATM address resolution, including inverse ARP requests.
- All of the IP-over-ATM network interfaces on a system must be members of the same logical IP subnetwork (LIS). However, no two members of the same LIS can reside within the same Origin processor module. This restriction is present only to enable automatic default and correct operation of the IRIX routing daemon (for example, `routed`). Nothing in the ATM protocol, the IRIS ATM software or hardware, or RFC 1577 requires this restriction. For RFC 1577 compliance, each IRIX ATM port can support one member of each LIS. For example, a module with 64 IRIX ATM ports could support 64 hosts from a single LIS. For ATM compliance, there are no restrictions regarding the number of logical network interfaces.

1.6.2 IP-over-PVCs in Compliance with RFC 1577

In environments requiring IP-over-PVCs in compliance with RFC 1577 (using or not using LLC/SNAP encapsulation and ATMARP), IRIS ATM can interoperate (that is, participate in each LIS) with or without ATM addresses and with or without ILMI.

1.6.2.1 Management Application for PVCs

Before IP traffic can be exchanged over PVCs, a network manager application must create the VCs and associate them with IP addresses. IRIS ATM ships the `atmarp` utility for this purpose. (Alternatively, a site can develop its own manager using the IRIS ATM application programming interface.) Once the PVC management application has created the PVCs, applications send and receive using the standard IRIX socket interface, just as with any other network subsystem that supports IP traffic.

1.6.2.2 PVC Management by `atmarp`

During system startup, the `/etc/init.d/network.atm` script starts the `atmarp` PVC management application if the `/var/atm/pvc.conf` IP-to-ATM

address resolution file exists. The user-configurable `pvc.conf` file maps IP addresses to *VC addresses*. (A VC address consists of a local port identifier and VPI/VCI values from the ATM cell). For each entry in the table, the `atmarp` daemon establishes a best-effort PVC and associates it with an IP address. The `atmarp` utility then goes to sleep, leaving the VCs open and ready for use. IP applications can then transmit and receive over the associated PVC. If `atmarp` is interrupted with a `SIGHUP` signal (for example, `killall -HUP atmarp`) it wakes up, reloads the lookup table from the `pvc.conf` file, makes any changes necessary, then goes back to sleep.

1.6.2.3 PVCs with LLC/SNAP Encapsulation

IP-over-PVCs, by default, operates with LLC/SNAP encapsulation and responds to inverse ATMARP requests. If a site wants to use ATM addresses, the ILMI module (`atmilmid`) must be configured, otherwise zero-length (null) ATM addresses are used. In a PVC environment, the ATM address is superfluous since the hardware connection is static: that is, (1) the PVCs are defined by the network manager as part of the system configuration; (2) they are created by the software (`atmarp` daemon) at startup time; and (3) they remain active until the network manager tears them down. The only endpoint address really needed is the IP address and the local VC address (VPI, VCI, and port tuplet) that are associated with each logical network interface.

The IRIS ATM subsystem handles PVC address resolution by using LLC/SNAP encapsulation with the following mechanisms:

- The IRIS ATM ILMI (`atmilmid`) module, at start up, automatically communicates with the adjacent switch on each ATM hardware connection (port) to create an ATM address for itself. On each ATM port that is using an ATM NSAP address, the `atmilmid` obtains the network-prefix portion of the ATM NSAP from the switch and registers its local portion (a MAC address). If the address request on any port times out without having received an address from the adjacent system, `atmilmid` uses the address configured in the `/var/atm/atmilmid.conf` file. If no ATM address is obtained from either source, a null source address is used.
- At startup, the IRIS ATM initialization script (`init.d.atm`) starts `atmarp`, which loads the contents of the `/var/atm/pvc.conf` file. This is the IP-to-VC address mapping table for PVCs. Each entry identifies one remote IP address, and maps it to a local hardware address consisting of a VPI/VCI address and a port identification number. The network portion of each IP address in this file must match the network portion of one of the logical network interfaces configured on this system; the software verifies that each entry has a local endpoint belonging to the same LIS.

- For each entry in the table, the `atmarp` daemon sets up both a transmit and a receive PVC. By default, each PVC is set up to use LLC/SNAP encapsulation, so that it supports IP inverse ARP protocol. Each PVC connects two members of an LIS, as shown in Figure 22, page 39.
- For each transmission request to one of these IP addresses, the IRIS ATM software looks in its IP-to-PVC address mapping table to discover the VCI/VPI and port (hardware) address on which to transmit. If no match is located, the transmission does not occur.
- For each received packet on any of these PVCs, the IRIS ATM software looks in its IP-to-PVC address mapping table to discover the local endpoint (IP address), then places that packet on the logical network interface's input queue.

  Note: In other words, each PVC to another IP address must be already set up and waiting before an application tries to send to that address and before packets start arriving at the switch from that address.
- If the IRIS ATM software receives an inverse ATMARP request, it responds with either the known ATM address or, if none is known, a zero-length address.

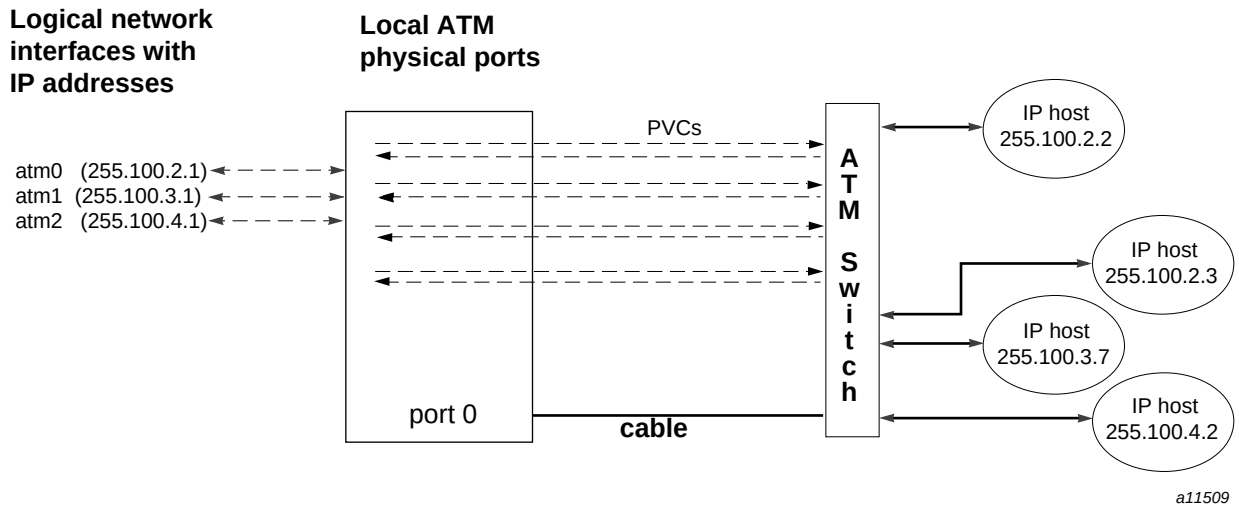


Figure 22. Multiple IP Interfaces Using a Single ATM Physical Port with PVCs

1.6.2.4 PVCs without LLC/SNAP Encapsulation

IRIS ATM allows IP to operate over PVCs without the overhead of IP ARP or LLC/SNAP encapsulation (in compliance with RFC 1577). This configuration functions exactly like IP-over-PVC with LLC/SNAP encapsulation (described in the previous section), except that the encapsulation is not used and inverse ARP replies are not generated. This functionality can be configured on a per-PVC basis in the `/var/atm/pvc.conf` file, as explained in Section 3.5.5.1, page 109.

The IRIS ATM subsystem handles PVC address resolution without LLC/SNAP encapsulation with the following mechanisms:

- At startup, the IRIS ATM initialization script (`network.atm`) calls the `atmarp` utility that loads the contents of the `/var/atm/pvc.conf` file into memory. This is the IP-to-PVC address mapping table. Each entry identifies one remote IP address, and maps it to a local hardware address consisting of a VPI/VCI address and a port identification number. The network portion of each IP address in this file must match the network portion of one of the logical network interfaces on this system in order to ensure that both endpoints belong to the same LIS.
- For each entry in the table, `atmarp` sets up both a transmit and a receive PVC. For non-LLC/SNAP usage, each entry must be marked in order to turn off this encapsulation. Each PVC connects two members of an LIS, as shown in Figure 22, page 39.
- For each transmission request to one of these IP addresses, the IRIS ATM software looks in its IP-to-PVC address mapping table to discover the VCI/VPI and port address on which to transmit. If no match is located, the transmission does not occur.
- For each received packet on any of these PVCs, the IRIS ATM software looks in its IP-to-PVC address mapping table to discover the local endpoint (IP address), then places that packet on the logical network interface's input queue.

  Note: In other words, each PVC to another IP address must be already set up and waiting before an application tries to send to that address and before packets start arriving at the switch from that address.

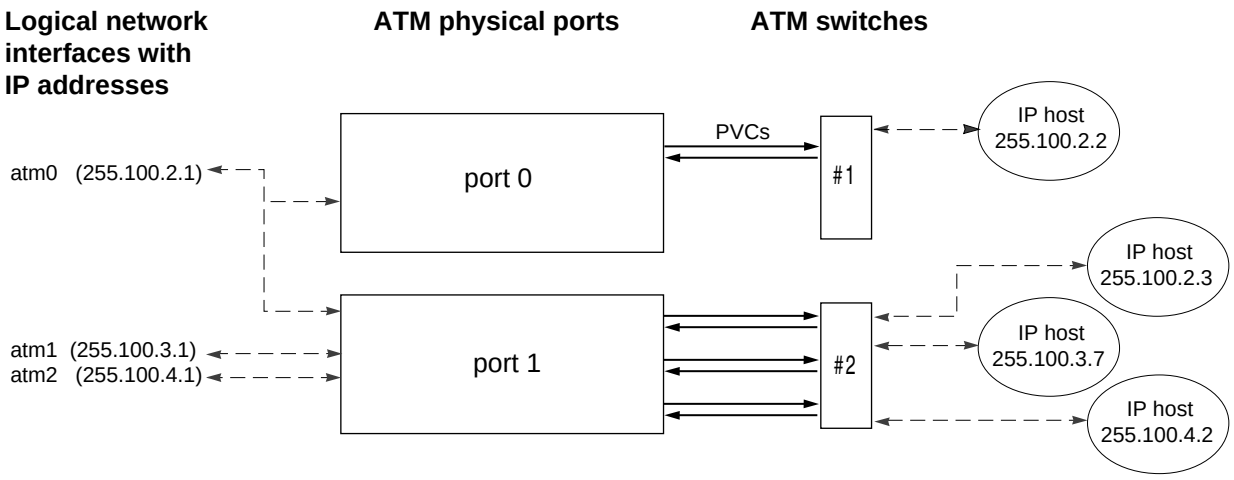
1.6.3 IP-over-PVC Configurations That Do Not Comply with RFC 1577

IRIS ATM allows IP to operate over PVCs in configurations that do not comply with RFC 1577 guidelines. This type of configuration functions exactly like the conformant IP-over-PVC configurations; however, the address resolution and

PVC management daemon, `atmarp`, cannot be used because it enforces RFC 1577 rules. The following scenarios (shown in Figure 23, page 42,) are examples of nonconformant configurations:

- Endpoints having IP addresses with the same network portion and subnet mask that cannot all contact each other directly. In example A, endpoints 255.100.2.2 and 255.100.2.3 cannot directly contact each other since they are physically connected to different ATM networks.
- An IP address is associated with more than one ATM address. In example A, endpoint 255.100.2.1 is known by two different ATM addresses (due to its use of two ports).
- IP addresses with the same network portion and subnet mask share a single virtual channel. In example B, multiple members of an LIS are sharing a single PVC.

Example A



Example B

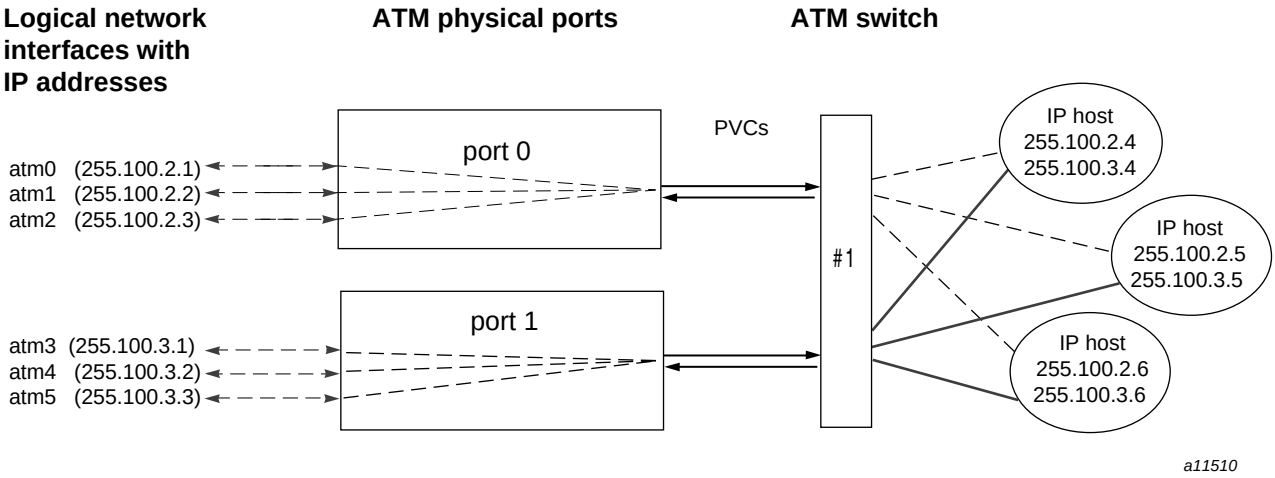


Figure 23. IP-over-PVC Configurations That Do Not Conform to RFC 1577

1.7 IRIS ATM Implementation Details

The following sections describe some of the implementation details for IRIS ATM products.

1.7.1 How IP Network Interfaces Are Assigned to ATM Boards

This section describes the manner in which the IRIX operating system assigns an IP network interface (for example, `atm0` and IP address 223.9.1.2) to a particular ATM port.

1.7.1.1 On CHALLENGE and Onyx Platforms

On a CHALLENGE or Onyx system, with each restart (for example, after a `reboot`, `shutdown`, `halt`, or `init` command, or a power off), the startup routine probes for hardware installed in the HIO mezzanine adapter slots, and makes a list of all the boards located. The slots are probed in the following order:

- Main IO4 board: I/O adapter slot 5, then 6
- Second IO4 board (if present): I/O adapter slot 2 (only when the FMezz board is long), slot 5, slot 3 (only when the FMezz board is long), and slot 6
- Third IO4 board (if present): I/O adapter slot 2 (only when the FMezz board is long), slot 5, slot 3 (only when the FMezz board is long), and slot 6
- Fourth IO4 board (if present): I/O adapter slot 2 (only when the FMezz board is long), slot 5, slot 3 (only when the FMezz board is long), and slot 6

The list and order of IRIS ATM boards that were located by this process can be displayed by using the `/sbin/hinv` command, as shown in the following example.

```
% /sbin/hinv -d atm
ATM-OC3c: atm0, slot 5 adap 6, firmware version ####
ATM-OC3c: atm1, slot 3 adap 5, firmware version ####
```

If the hardware unit numbers are assigned by software (the default behavior), the text `atm#` indicates the order, as follows: `atm0` (also known as `unit0`) is the first board located and `atm1` is the second. If the unit numbers are set by jumpers, the text `atm#` represents the unit number read from the setting on the board. In the preceding example, the startup routine located two IRIS ATM boards attached to two different IO4 boards.

As the startup process begins to initialize ATM network interfaces, it does the following:

- If the IRIS ATM driver is configured to support the IP protocol stack, the driver creates the number of IP-over-ATM network interfaces specified in the `/var/sysgen/master.d/if_atm` file.
- The `/etc/init.d/atm` script uses the contents of the `/var/atm/ifatm.conf` file to associate each network interface with a board.
- The `ifconfig` command (which is invoked automatically during startup) searches the `netif.options` file for IP-over-ATM interface names (for example, `atm0`, `atm1`, `atm2`) and configures and enables each interface that exists (that is, each interface that was created by the driver).

1.7.1.2 On Origin and Onyx2 Platforms

On an Origin2000, OriginServer, or Onyx2 system, with each restart (for example, after a `reboot`, `shutdown`, `halt`, `init` command, or a power off), the startup routine probes for hardware on all the systems connected into the Craylink interconnection fabric. All the slots and links in all the modules within the fabric are probed. The routine then creates a hierarchical file system, called the hardware graph, that lists all the hardware that is located. The top of the hardware graph is visible at `/hw`. (For complete details, see the man page for `hwgraph`.) After the hardware graph is completed, the `ioconfig` program assigns a unit number to each located device that needs one. Other programs (for example, `hinv` and the device's driver) read the assigned number and use it. On an initial startup, `ioconfig` assigns numbers sequentially; for example, if two IRIS ATM XIO boards are found, they are numbered `unit0` (with ports 0 to 3) and `unit1` (with ports 4 to 7). The port numbers for each IRIS ATM XIO board are created from the board's assigned unit number: *first_port#=brd_unit#* * 4 and the other three are sequential from the first. On subsequent startups, `ioconfig` distinguishes between hardware that it has seen before and new items. To previously seen items, it assigns the same unit and port numbers that were assigned on the initial startup. To new hardware, it assigns new sequential numbers. It never reassigns a number, even if the device that had the number is removed and leaves a gap in the numbering.

Note: New items are differentiated from previously seen items based solely on the hardware graph listing (that is, the path within `/hw`). The database of previously seen devices is kept in the file `/etc/ioconfig.conf`. For example, a replacement board that is installed into the location of an old board will be assigned the old board's numbers, while a board that is moved from one location to another will be assigned a new unit number and new port numbers. For more information about the hardware graph and `ioconfig`, see the man pages for `hwgraph` and `ioconfig`.

The IRIS ATM boards that are located can be displayed with the `/sbin/hinv` or `find` commands, as shown in the following examples. In these examples, the startup routine located two IRIS ATM boards on two different modules (that is, inside two different chassis).

```
% find /hw/module -name atm
/hw/module/1/slot/io3/quad_atm/pci/0/atm
/hw/module/2/slot/io12/quad_atm/pci/0/atm<

%   /sbin/hinv -d atm
ATM XIO 4 port OC-3c: module 1, slot io3, unit 0 (ports: 0-3)
ATM XIO 4 port OC-3c: module 2, slot io12, unit 1(ports: 4-7)
```

As the startup process continues, it calls the network hardware drivers so that they can create their network and programmatic interfaces. For ATM, this step works in the following manner:

- For each IRIS ATM port, the startup process creates short (`/hw/atm/#`) and long (`/hw/module/#/slot/.../atm.`) entries in the hardware graph. Then, the installation scripts create a symbolic link in `/dev` that points to the port's entry in the hardware graph. The `/dev/atm#` links are for use by the IRIS ATM application programming interface (API).
- If the IRIS ATM driver is configured to support the IP protocol stack, the driver creates the number of IP-over-ATM network interfaces specified in the`/var/sysgen/master.d/if_atm` file.
- The `/etc/init.d/atm` script uses the contents of the `/var/atm/ifatm.conf` file to associate each IP-over-ATM network interface with a port.
- The `ifconfig` command searches the `netif.options` file for IP-over-ATM interface names (for example, `atm0`, `atm1`, `atm2`) and configures and enables each interface that exists (that is, each interface that was created by the driver).

1.7.2 How Transmission Rates Are Managed

The following sections describe how transmission rates are created and managed by different IRIS ATM hardware products.

1.7.2.1 Rates on the ATM-OC3c HIO Board for CHALLENGE and Onyx Platforms

The IRIS ATM-OC3c HIO board for CHALLENGE and Onyx platforms manages transmission rates with rate queues and divisors. The board has 8 rate queues organized as 2 banks: a0-a3 and b0-b3. Each queue can support one peak rate and 63 different sustainable rates. The "a" bank consists of 4 high-priority queues that are designed for constant bit rate traffic (CBR and VBR channels). The other bank contains 4 low-priority queues and are only used for best-effort traffic.

High-priority queues are serviced before low-priority ones. As long as there is data awaiting transfer on any high-priority queue, low-priority data is not transmitted. This means that, for applications with a constant flow of data, only queues a0-a3 will ever operate.

During startup, the IRIS ATM HIO board driver configures each rate queue, as follows:

- Queues that are mentioned in the `/var/atm/atmhw.conf` file are configured to a fixed rate, as specified in the file. The IRIS ATM driver never changes the rates for these queues; this ensures that site-specified rates are always available, even when the queues are not actively being used. Appendix B, page 211, lists the supported rates, which range from 0 to 135,991,460 bits-per-second.
- Queues that are not mentioned (or are commented out) in the file are left unconfigured. The driver configures these during operation, as follows.

During operation, as VCs are created, the driver associates each newly created VC with the queue whose transmission rate best matches the peak rate requested for that VC. For each `ATMIOC_CREATEPVC` or `ATMIOC_SETUP` command, the driver looks for a queue whose transmission rate best matches the rate requested in the API call, following these guidelines:

- For VCs carrying best-effort traffic, the driver uses the low-priority queue whose rate is closest to, but slower than, the requested peak rate.
- For VCs carrying CBR and VBR traffic, the driver uses the high-priority queue whose configured rate exactly matches the requested peak rate. If the requested rate does not exist, the driver searches for a high-priority queue

with the following characteristics and reconfigures it to the requested peak rate:

- A queue that does not currently have a VC associated with it
- A queue that was not configured from the `atmhw.conf` file during startup

  Note: There can be dozens of CBR and VBR virtual channels active on a board, but the peak rate for each one must be one of the four rates that are configured on the high-priority queues.

To set the sustainable transmission rate for a particular VC, one of the board's configured rates is divided by a divisor (ranging between 1 and 64). The IRIS ATM driver sets all divisors. Peak rates for CBR, VBR, or best-effort traffic use divisors of 1. Sustainable (average) rates for VBR traffic use divisors between 2 and 64 (inclusive).

Note: The IRIS ATM-OC3c board for CHALLENGE and Onyx platforms implements sustainable (average) rates by filling some cell slots with idle cells instead of data. This results in some loss of bandwidth which, under heavy load conditions, can be noticeable.

To summarize, the IRIS ATM-OC3c board simultaneously makes available for selection up to 8 different peak rates and up to 504 (8*63) sustainable rates. Not all of these available selections can be actively used simultaneously, since this would exceed the board's bandwidth.

Table 5, page 47, summarizes the default settings configured for the IRIS ATM-OC3c board's rates.

Table 5. Default Transmission Rates on ATM-OC3c Queues

Rate queue Number ID	String ID	Default cell rate (in ATM cells per second)	Default bit rate (in user payload bits per second)	Priority / Use
0	`a0`	unconfigured	0	High / CBR, VBR[13]
1	`a1`	unconfigured	0	High / CBR, VBR
2	`a2`	unconfigured	0	High / CBR, VBR

Rate queue Number ID	String ID	Default cell rate (in ATM cells per second)	Default bit rate (in user payload bits per second)	Priority / Use
3	`a3`	unconfigured	0	High / CBR, VBR
4	`b0`	26,041	10,000,000	Low / BE
5	`b1`	78,125	30,000,000	Low / BE
6	`b2`	178,571	68,000,000	Low / BE
7	`b3`	357,142	135,991,460	Low / BE

A board is oversubscribed when the sum of all the open VCs multiplied by their average rates is greater than the board's total payload bandwidth.[14] The IRIS ATM software contains a number of features that prevent performance degradation due to oversubscription.[15] Whenever there is even one VC open for a CBR traffic contract, the IRIS ATM software refuses to create new VCs once the board's total bandwidth is allocated to open VCs (including the best-effort ones). If all the VCs on a board are best-effort (regardless of which queues they are using), the IRIS ATM software allows the board to become oversubscribed and handles the transmissions in the best manner possible.

Note: With this board, the default TCP/IP configuration uses the maximum bandwidth possible. Therefore, a single TCP/IP connection can oversubscribe the port it uses and prevent CBR traffic. To prevent this, there are two options: (1) reduce the default TCP/IP bandwidth (for example, by editing the `/var/atm/ifatm.conf` file) or (2) use `ifconfig` to disable the TCP/IP logical network interfaces.

13 CBR = constant bit rate; VBR = variable bit rate; BE = best effort

14 Total OC3c bandwidth is 155.52 megabits per second; however, of this, about 87.2% (135,631,698 bits) is available for upper–layer data. This is referred to as the payload bandwidth.

15 When a VC specifies a sustainable rate, this is the average rate. When the VC does not specify a sustainable rate, the peak rate is used as the average rate.

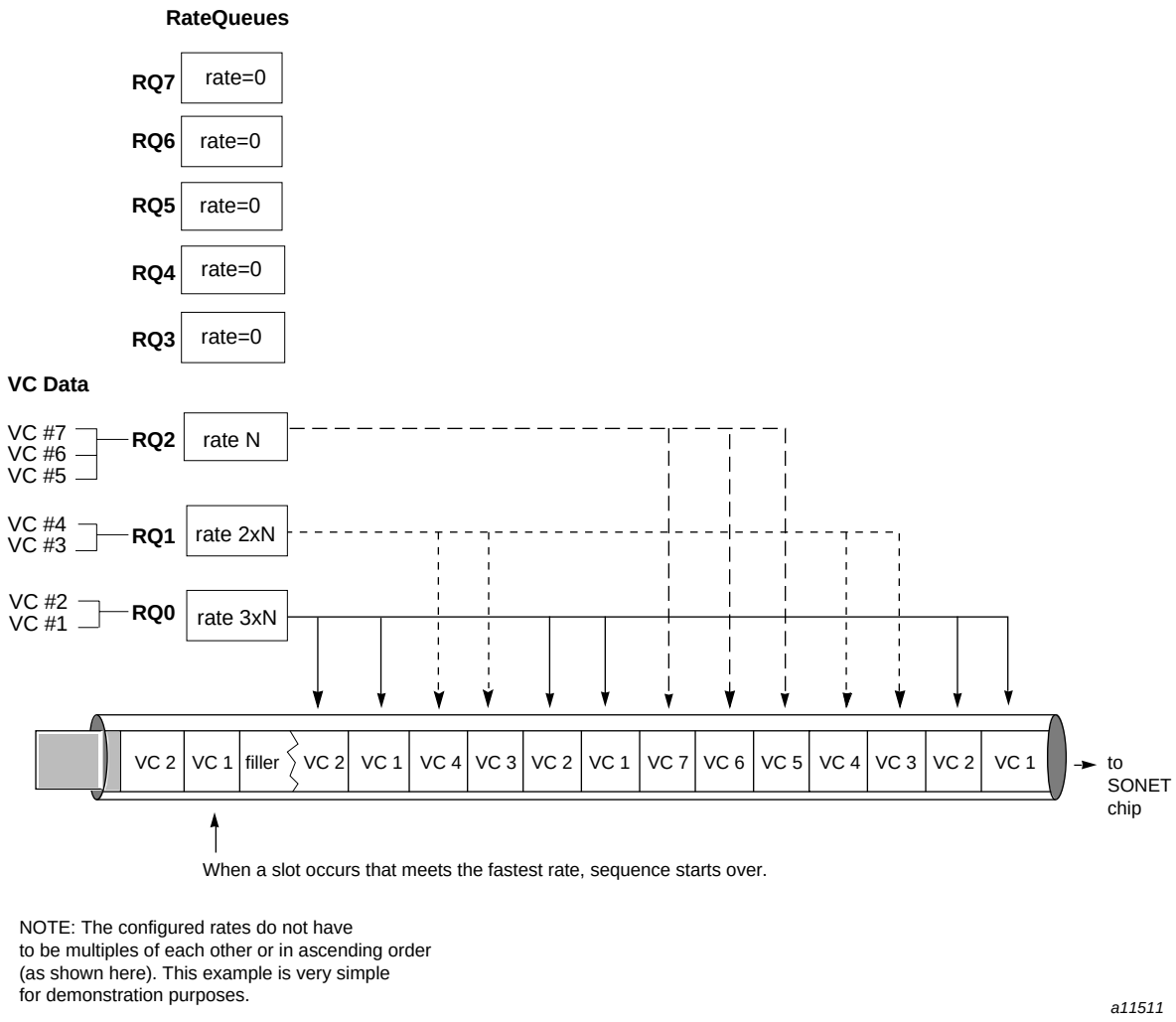


Figure 24. Transmission Rate Control on ATM-OC3c Board for CHALLENGE and Onyx Platforms

1.7.2.2 Rates on the ATM-OC3c XIO Board for Origin2000 and Onyx2 Platforms

The IRIS ATM-OC3c 4Port XIO board for Origin2000 and Onyx2 platforms handles transmission rates with a table that is managed by firmware. Each table entry represents one ATM cell slot on the output stream, as shown in Figure 25, page 51. For each VC, the board's firmware calculates how many table entries are needed to generate the VC's rate, then associates that number of table

entries with the VC's data stream, as shown in Figure 25, page 51. The cell feeder logic sequentially moves through the table, entry by entry. Each entry triggers a 48-byte read (384 bits) from one data stream. An ATM cell is created and placed on the stream of data that is on its way to the SONET logic. If there is no VC associated with a table entry, the cell feeder creates an idle cell to fill that cell slot in the stream. As shown in Figure 25, page 51, a VC is associated with multiple, evenly spaced table entries in order to transmit the VC's data steadily at its transmission rate. For this board, a VC's sustainable rate is exactly the same as its peak rate.

Theoretically, for a table with 353,207 entries, each entry would be processed once every second, while the entries in a table with half that number of entries would be processed twice every second. To understand why this is true, see the following calculation and facts:

- The OC3c SONET line rate is 155, 520,000 bits per second of which approximately 87.2% (135,631,698 bits) is upper-layer payload, and the rest is SONET and ATM overhead.
- The upper-level data in each ATM cell (and hence, each entry in the table) places 384 bits onto the transmission medium.
- When the total number of upper-layer bits transmitted every second is divided by the upper-layer bits in each cell, the result is the number of ATM cells per second. This is the number of table entries that are processed once every second: 135,631,698 bits transmitted per second / 384 bits per cell= 353,207 cells per second.

  Or expressed another way, the (number of table entries) multiplied by (number of times each table entry is processed every second) multiplied by (upper-layer bits in each cell) equals the total number of upper-layer bits transmitted in a second: (353,207 * 1) * 384 = 135,631,408 maximum bits per second of payload (176,603 * 2) * 384 = 135,631,108 maximum bits per second of payload.

A more realistic example for this product's hardware implementation (in which the maximum number of table entries is limited) is the following example:

(7,064 entries * 50 table passes) * 384 bits per cell = 135,628,808 payload bits per second

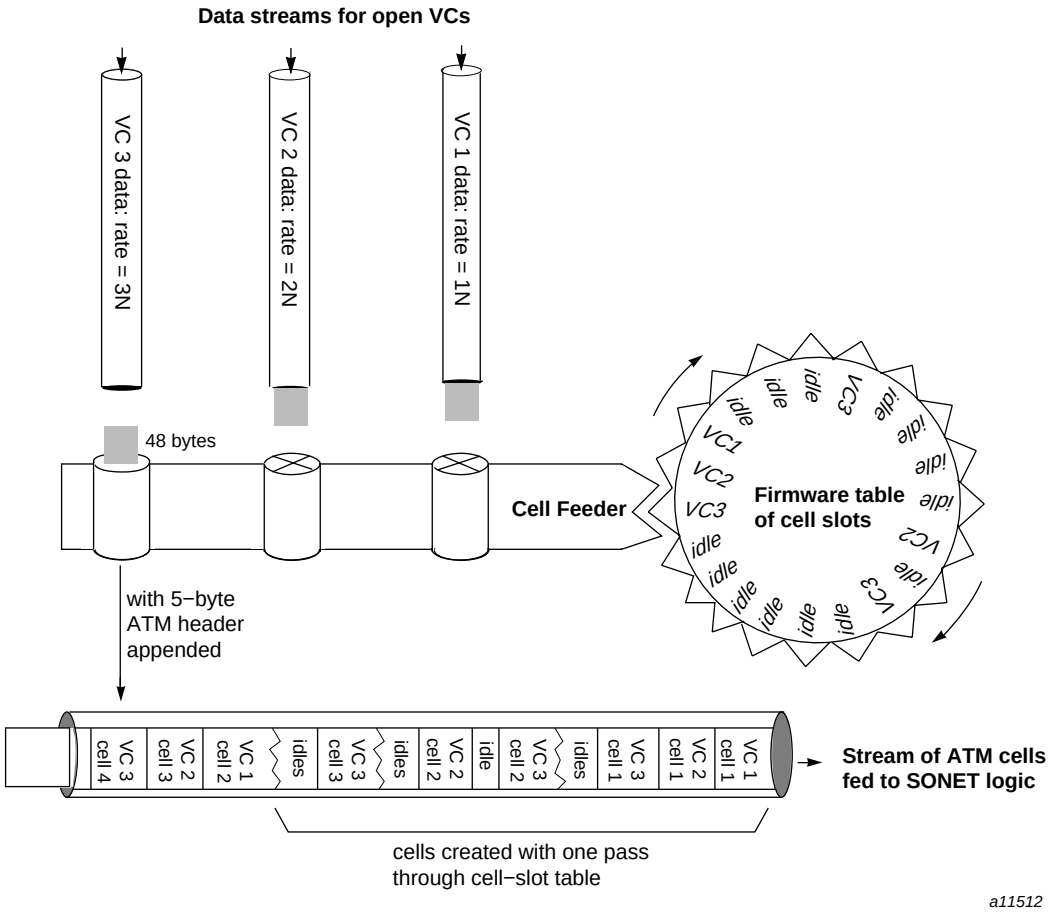


Figure 25. Transmission Rate Control on ATM-OC3c XIO Board for Origin2000 and Onyx2 Platforms

The driver for the IRIS ATM-OC3c 4Port XIO board does not allow oversubscription of any port. Whenever a requested rate cannot be provided, the request is denied. A denial can be due to either of these reasons:

- Not enough table entries are free to create the requested transmission rate. Said another way, the request is denied when filling it would oversubscribe the line rate.
- The spacing possible with the currently available table entries is not even enough to create a steady data flow for the VC. For example, if the table entries for a requested rate need to be spaced at intervals of 50 (table entries

8, 58, 108, 158, and so on, or table entries 3, 53, 103, 153, and so on), the request is denied when one or more of the needed table entries are already filled. This denial can occur even though the requested rate does not oversubscribe the connection's line rate.

1.7.3 Support for Upper Layer Applications

IRIS ATM supports the following upper layer applications:

- Standard IRIX TCP/IP applications:

  For Internet (IP) networking, IRIS ATM provides its services to the IRIX IP protocol stack. IP applications can use the IP–over–ATM logical network interfaces (`atm#`), just as they would IP over Ethernet or FDDI. This support provides RFC1577-compliant address resolution and packet encapsulation. IP traffic can be exchanged over dynamically-created switched virtual circuit (SVC) connections or permanent virtual circuit (PVC) connections.

  With SVCs, IP-to-ATM address resolution is handled by an ATMARP server (as specified by RFC1577). With PVCs, IP-to-ATM address resolution is handled by the IRIS ATM `atmarp` daemon.

  With SVCs, the creation of channels is handled by the IRIS ATM `atmsigd` module which provides a private user-to-network interface (UNI) as specified in the official standard: *ATM User-Network Interface Specification, Versions 3.0 and 3.1* (ATM UNI). With PVCs, the creation of channels is handled by the IRIS ATM `atmarp` daemon.

  For both SVCs and PVCs, the `atmilmid` module provides interim local management interface (ILMI) support and address assignment and registration, as specified in the ATM UNI standard.

- IRIS ATM utilities:

  IRIS ATM includes utilities (for example, `atmconfig`, `ifatmconfig`, `atmstat`, `sigtest`, and `atmtest`) for configuring, monitoring, and testing the IRIS ATM subsystem.

- Customer-developed applications:

  IRIS ATM provides an application programming interface (API) that customers can use to develop their own upper-layer applications that use PVCs or SVCs for IP or non-IP traffic. See the *IRIS ATM API Programmer's Guide* (shipped with each IRIS ATM board) for details.

Initial Configuration and Verification [2]

This chapter explains the procedures that should be used to configure and verify IRIS ATM the first time it is started on a system. For subsequent configuration changes and for details of the procedures explained in this chapter, see the instructions in Chapter 3, page 81.

The configuration steps in this chapter configure the product with default settings that allow it to be tested with the verification procedures described at the end of the chapter. The configuration that results from these procedures is a default. In many environments, the site's network administrator will need to alter the settings after the product has been verified.

2.1 Order for Performing Installation Steps

It is highly recommended that you use the following order to install and configure the IRIS ATM product:

1. Use `inst` to install the new IRIS ATM software; the software image is named `atm`. Use of `inst` is explained in the *IRIS ATM Release Notes* and the IRIS Insight document *Software Installation Administrator's Guide*.
2. Configure IRIS ATM, as explained in Section 2.3, page 57.
3. If not already done during the configuration steps, invoke the `/etc/autoconfig` command to build the IRIS ATM driver into the operating system.
4. Power down the system and install the IRIS ATM hardware, as explained in the *IRIS ATM\x15 OC3c Board for CHALLENGE and Onyx Installation Instructions* or *IRIS 4Port ATM\x15 OC3c XIO Board Installation Instructions*.
5. Power on the system.
6. Verify the IRIS ATM subsystem, as explained in Section 2.4, page 60.

Note: Failure to follow this order of installation steps will require additional reboots of the system. The hardware cannot be verified until the software is installed, configured, booted twice (or autoconfigured by using `autoconfig`, and booted).

2.2 Before You Begin Configuration

Before you begin the configuration, you should perform the following steps:

1. Determine which functionalities your system needs to support.
2. Collect configuration information.
3. Review required configuration tasks.

The following sections offer guidelines for performing these steps.

2.2.1 Determining System Functionalities

This product can be configured to support any or all of the functionalities listed in this section. Before you begin the configuration, you should determine which of the following functionalities your system needs to support:

- IP applications over switched virtual circuits (SVCs) in compliance with RFC 1577
- Non-IP applications (developed by the customer) over SVCs
- IP applications over permanent virtual circuits (PVCs)
- Non-IP applications (developed by the customer) over PVCs

2.2.2 Collecting Configuration Information

Before starting the installation and configuration, collect the following information that is required during the configuration procedures. Space is provided in this section for you to record the information.

After you have collected the configuration information, follow the steps in Section 2.3, page 57, to configure the IRIS ATM subsystem.

1. Will this system use the IP (also called TCP/IP) protocol over any of its ATM ports? If yes, determine how many IP-over-ATM logical network interfaces (`atm#`) are needed and obtain an IP address for each one. For IP-over-ATM, it is possible to configure multiple interfaces per physical port; specifically, there can be one interface for each logical IP subnetwork (LIS). The ATM subsystem can support up to 64 logical network interfaces.

`atm0`_____

`atm`_____

`atm`_____

`atm`_____

2. Will this system use PVCs? If it will, obtain the IP address and a VPI/VCI value for one or more ATM endpoints with which this system will be communicating. Also determine which port number (as displayed by the `hinv` command) will support each of these channels.

   _____ / _____port__

   _____ / _____port__

   _____ / _____port__

   _____ / _____port__

   If any of these PVCs needs LLC/SNAP encapsulation disabled, mark that PVC's line with an `n`. In compliance with RFC 1577, IRIS ATM does LLC/SNAP encapsulation and Inverse ARP responses by default. IRIS ATM does not generate Inverse ARP requests.

3. Will this system use SVCs? If it will, obtain the following information:

   - For each IP-over-ATM logical network interface (`atm#`) that will use SVCs, obtain the ATM address (either ATM NSAP or native E.164) of the ATM address resolution server:

     `atm`_____

     `atm`_____

     `atm`_____

     `atm`_____

   - For each physical port (number displayed by `hinv`), find out which version of the ATM UNI (3.0 or 3.1) the attached switch supports for service specific convergence protocol (SSCOP) and Q.2931:

     port0SSCOP=____Q.2931=____

     port1SSCOP=____Q.2931=____

port2SSCOP=____Q.2931=____

port3SSCOP=____Q.2931=____

- For each port, does the adjacent switch perform ILMI address registration, or will you need to assign the system its ATM address on that port? (There should be one switch for each IRIS ATM physical port.) If all the switches do address registration, skip to the next step. For ports that you need to assign the ATM address, obtain either a native E.164 or an ATM NSAP address.

  port0_____

  port1_____

  port2_____

  port3_____

  Note: If no switch is attached to a port, follow the instructions in Section 3.3.4.4, page 100.

2.2.3 Reviewing Required Configuration Tasks

Table 6, page 56, lists the configuration tasks that must be performed after the new IRIS ATM software has been installed, but before IRIS ATM is started for the first time and its functionality verified. The IRIS ATM subsystem cannot function until these tasks have been completed. Brief instructions for all of these tasks are provided in Section 2.3, page 57.

Table 6. Required Configuration Tasks

	Item to configure	Section where complete instructions are located
For IP traffic over ATM:		
	IP addresses and names	Section 3.5.2, page 106
	Network interface to IP address mappings	Section 3.5.3, page 106
	ATM-to-IP address mappings (address resolution)	Section 3.5.5, page 109

	Item to configure	Section where complete instructions are located
For SVCs only:		
	ATM UNI and signaling	Section 3.6.1, page 123
	ILMI module	Section 3.7.1, page 126

After these required configuration tasks have been performed, the IRIS ATM subsystem operates with default settings, some of which may not be suitable for your purposes. After performing the verification tests, if you need to alter the default parameters, follow the more complete instructions provided in Chapter 3, page 81, to configure the system with different settings.

2.3 Configuring IRIS ATM

Follow the procedures in this section to configure the IRIS ATM product the first time it is installed and before it is run. The product is not operational until it is configured as described in this section.

1. Locate the `inst` image for the new version of IRIS ATM. Then, use `inst` to install it, as illustrated in the following command lines:

   ```
   # inst
   Inst> from location_of_image
   Inst> install atm
   Inst> go
   ```

2. For systems requiring more than four IP-over-ATM network interfaces, edit the `/var/sysgen/master.d/if_atm` file to configure the number of ATM logical network interfaces needed. As shown in the following example, change the `ifatm_n_ifnets=` line to replace the default value (4) with a decimal number to indicate the number of interfaces your system requires. Complete instructions are provided in Section 3.4, page 102.

   Change from this:

   ```
   ifatm_n_ifnets=4
   ```

   To this:

   ```
   ifatm_n_ifnets=decimal_number
   ```

3. For systems requiring IP-over-ATM, edit the `/etc/hosts` file to configure an IP address and name for each ATM logical network interface. Complete instructions are provided in Section 3.5.2, page 106.

4. For systems requiring IP-over-ATM, edit the `/etc/config/netif.options` file to map each network interface (including `atm0`) to an IP address. Complete instructions are provided in Section 3.5.3, page 106.

5. For CHALLENGE or Onyx systems (only) exchanging constant bit rate (CBR) traffic, if any particular rates need to be always available, edit the `/var/atm/atmhw.conf` file to configure these permanent rates. Complete instructions are provided in Section 3.3.3.2, page 95.

6. For systems using IP–over–PVCs, edit the `/var/atm/pvc.conf` file to map each remote endpoint (IP address) to an ATM virtual channel address and an ATM port. If the channel does not use LLC/SNAP encapsulation, place an `n` in the flags column. Each entry should have the following format. Complete instructions are provided in Section 3.5.5.1, page 109.

   Format:

   IPaddress port# vpi vci fflags

   Example:

   ```
   223.16.8.1  0  1  35
   ```

   Note: You do not need to add the PVCs used by ATM signaling and ILMI. The IRIS ATM software does this automatically.

7. For systems using IP-over-SVCs, edit or create a `/var/atm/ifatm.conf` file to map each IP-over-SVC logical network interface to an ATM port and an ATM address resolution server. Each entry should have the following format. Complete instructions are provided in Section 3.5.5.2, page 111.

   Format:

   `atm# port # arpserver` *NSAP_address*

   Example:

```
atm0 port 0 arpserver 0x39084000112233445566778899 0d000122003300
```

Note: To make this station operate as the ATMARP server for the LIS in which `atm#` is a member, add the first two items in the line but do not specify `arpserver`. After completing the configuration and rebooting the system, edit this file again to add the `arpserver` portion of the line, as described in Section 3.5.5.2, page 111.

8. For systems using SVCs, edit the `/var/atm/atmsigd.tcl` file to configure the UNI versions used on each port. The versions (for SSCOP and Q.2931 signaling) must match those used by the adjacent switch. For each port, remove the pound sign (#) from one line in the file or create new lines, as shown in the following example. Complete instructions are provided in Section 3.6.1, page 123.

   Format:

   `buildstack` *identification*`#` `/hw/atm/#` *version_SSCOP version_Q.2931*

   Change from this:

```
buildstack 1 /hw/atm/0 AF30 AF30
# buildstack 1 /hw/atm/0 AF30 AF31
# buildstack 1 /hw/atm/0 AF31 AF31

# buildstack 2 /hw/atm/1 AF30 AF30
# buildstack 2 /hw/atm/1 AF30 AF31
# buildstack 2 /hw/atm/1 AF31 AF31

# buildstack 3 /hw/atm/2 AF30 AF30
# buildstack 3 /hw/atm/2 AF30 AF31
# buildstack 3 /hw/atm/2 AF31 AF31
```

   To something like this:

```
buildstack 1 atm0 AF30 AF30
# buildstack 1 atm0 AF30 AF31
# buildstack 1 atm0 AF31 AF31

# buildstack 2 atm1 AF30 AF30
buildstack 2 atm1 AF30 AF31
# buildstack 2 atm1 AF31 AF31

buildstack 3 atm2 AF30 AF30
# buildstack 3 atm2 AF30 AF31
# buildstack 3 atm2 AF31 AF31
```

```
# buildstack 4atm3 AF30 AF30
# buildstack 4atm3 AF30 AF31
buildstack 4atm3 AF31 AF31
```

9. For systems using SVCs, edit the `/var/atm/atmilmid.conf` file to configure the ILMI module for each physical port (that is, each ATM-OC3c board). For any port attached to a switch that does not register ATM addresses, also add an `ATMADDRESS` line to assign the system its ATM address. Complete instructions are provided in Section 3.7.1, page 126.

   Format:

   `ATMPORT` *port_index devname VPI VCI*
   `ATMADDRESS` *port_index ATM_address*

   Example:

```
ATMPORT  1  /hw/atm/0  0  16
ATMPORT  2  /hw/atm/1  0  16
ATMPORT  3  /hw/atm/2  0  16
ATMPORT  4  /hw/atm/3  0  16
ATMADDRESS 1 47000580ffe1000000f115098d080069042a4f00
```

10. Invoke the `/etc/autoconfig -f` command to build the IRIS ATM driver into the operating system that will be loaded at the next startup.

The system is now ready to shut down and install the hardware. If the hardware is already installed, reboot the system now. Once the hardware is installed and the system is powered on, follow the instructions in Section 2.4 to verify that the ATM subsystem is functional.

2.4 Verifying IRIS ATM Functionality

The following sections describe procedures for verifying the functionality and configuration of an IRIS ATM subsystem. Do all of the procedures described in each section that are appropriate for your configuration, as described in the introductory paragraph for each set of verification steps.

2.4.1 Verifying the Initial Configuration

This procedure verifies that the basic configuration information about the IRIS ATM subsystem is correct. Do these steps immediately after the initial installation, configuration, and startup. Skip any steps intended for a configuration that does not apply.

1. Log on to the system and become super user (root).

2.4.1.1 Verify the Board Configuration

2. Use `hinv` to verify that the IRIS ATM board has been located by the operating system, as follows:

```
# /sbin/hinv
. . .
```

In a CHALLENGE or Onyx system (HIO board), the format of the `hinv` display is as follows:

`ATM OC-3 unit` *unit*`#: slot` *slot*`#, adapter` *adapter*`#`

The *unit*# option indicates the board's (port's) unit number, *slot*# indicates the slot in which the IO4 board resides, and *adapter*# (IO4 adapter) indicates the mezzanine position (5 indicates lower; 6 indicates upper) at which the ATM board resides.

Note: If the board is not listed, either (1) the system is running the wrong IRIX version, (2) the system has not been booted enough times, (3) the jumpers on the newly installed board have the same unit number as one of the listed ATM boards, or (4) the board is improperly installed. If no ATM board is listed, use the `versions eoe1` command to verify the IRIX version number. If the version is incorrect, install the correct version, and reboot the system two times. If the version is correct or if other IRIS ATM boards are listed, reboot and watch the terminal for error messages. If the problem persists, reinstall the product (the board and all cables) making sure to set the IRIS ATM board's jumpers correctly and to set all the hardware firmly.

In an Origin2000 or Onyx2 system, the format of the `hinv` display is as follows:

`ATM XIO 4 port OC-3c: module` *module*`#, slot` *slot*`#, unit` *unit*`# (ports: #-#)`

The *module*# and *slot*# variables indicate the location (module and XIO slot) at which the board resides, *unit*# indicates the board's assigned unit number, and #-# indicates the range of numbers assigned to the four ports.

Note: If the board is not listed, either (1) the system is running the wrong IRIX version, (2) the IRIS ATM driver has not been built into the IRIX system by using the `autoconfig` command, or (3) the board is improperly installed. If no ATM board is listed, use the `versions eoe` and `versions atm` commands to verify the IRIX and IRIS ATM versions. If either image is incorrect or missing, install the correct version, and reboot the system. If the version is correct or if other IRIS ATM boards are listed, reboot and watch the terminal for error messages. If the problem persists, reinstall the product, making sure to set all the hardware firmly.

If the IRIS ATM board is listed, continue to the next step.

3. Use `atmconfig` as follows to verify that each port's MAC address is recognized:

```
# /usr/etc/atmconfig -i port# -m
MAC addr: ##:##:##:##:##:##
```

The *port#* indicates the port's identification number. For the HIO mezzanine board in a CHALLENGE or Onyx system, the *port#* is the same as the board's unit number.

If the MAC address is listed, continue to the next step.

Note: If the command was not located, the IRIS ATM software is not installed. If `atmconfig` displayed a message, look the message up in Chapter 5, page 153. If the display shows NULL for a MAC address, the IRIS ATM board is dysfunctional. Contact the Silicon Graphics Technical Assistance Center or the site's support engineer.

4. On a CHALLENGE or Onyx system, use `atmstat` as follows to verify that the board's rate queues are configured:

```
# /usr/etc/atmstat -i unit# -q
/hw/atm/#: interface HW state: UP
ATM interface rateq settings:
0: #### cells/s (#.## Mbps)
. . .
```

If the rates are listed and are correct, continue to the next step. The rates that are configured by default are listed in Table 12, page 95.

Note: If the command was not located, the IRIS ATM software is not installed. If `atmstat` displayed a message, look the message up in Table 12, page 95. If the displayed rates are not correct, reconfigure them as explained in Chapter 3, page 81.

2.4.1.2 Verify IP Configuration

5. Use `netstat` as follows to verify that the IRIS ATM logical IP network interfaces are configured and enabled:

   ```
   % /usr/etc/netstat -ina
   . . .
   atm0 9180 netaddress hostaddress . . .
   ```

   If the IRIS ATM logical network interfaces are listed and enabled, continue to the next step.

   Note: If no ATM logical network interface is listed, there is something wrong with the configuration of the software. Use the `versions` command to verify that the IRIS ATM software is installed, then use `autoconfig` and reboot one more time, in case you did not start using the newly built operating system that contains the IRIS ATM software. If an ATM interface is listed, but not configured correctly, verify your entries to the following files:

   ```
   /etc/hosts
   /etc/config/netif.options
   /etc/config/ifconfig-#.options
   /var/sysgen/master.d/if_atm
   ```

   If an interface that you expected is not listed, verify your entries in the `/var/sysgen/master.d/if_atm` file.

2.4.1.3 Verify IP-over-PVC Configuration

6. Use `atmarp` as follows to verify that the PVCs are loaded into the address resolution table. Entries for PVC connections have the flag PVC.

   ```
   % /usr/etc/atmarp -a
   IP address port VPI VCI flags
   name         0    #  ##   PVC
   name         0    #  ##   PVC NOSNAP
   ```

   If there are no entries, follow the instructions in Section 3.5.5.1, page 109, to create and load entries.

2.4.1.4 Verify IP-over-SVC Configuration

7. Use `ps` as follows to verify that the ATM signaling and ILMI software are operating:

```
% /sbin/ps -ef | grep atm
root ... /usr/etc/atmsigd -d
root ... /usr/etc/atmilmid -p
```

   If the two daemons are listed, continue to the next step.

   Note: If the either or both daemons are not listed, one of the following problems might be the cause: (1) the missing module is not configured correctly or (2) the module was unable to contact the adjacent switch, so it terminated itself; this could indicate that the physical link is broken or missing, or that the switch is down. Solve the problem, then restart the daemons by following the instructions in Section 3.9, page 131.

8. Use `atmstat` to verify that `atmsigd` and `atmilmid` each have opened their PVC for use in their own communications. The VPI/VCI values should match either the defaults (0/5 and 0/16) or the values given during the configuration process.

```
% /usr/etc/atmstat -i port# -V
VPI VCI rateQ/div/burst   flags
 0   16  B0 / 2 / 32     READ WRITE
 0    5  B0 / 2 / 32     READ WRITE
 <other VCs>
 number receive VCCs: 4   number forward VCCs: 4
```

9. Use `ifatmconfig` as follows to verify that the ATM addresses are known and that the LIS information is configured:

```
% /usr/etc/ifatmconfig atm#
atm#: (ATM addr: 0xATM_address)
port #
arpserver 0xATM_address
vcrate #bps bps
vctimeout #minutes minutes
```

   If the all information is listed and correct, go to the next step.

Note: If an ATM address is not listed, use `grep atm` to search the SYSLOG file for `atm` error messages, then look them up in Chapter 5, page 153. Common causes for a missing ATM address include (1) the adjacent switch does not support address registration or (2) the `/var/atm/ifatm.conf` file could not be parsed during startup or did not contain an entry for the logical network interface.

10. Use `atmarp` to verify that the SVC to the ATMARP servers are open. Connections to ARP servers can be identified by the server's ATM address, as shown in the following example. The ATM address shown here should match the one displayed for `arpserver` in the previous step. The IP address for an ATMARP server is not necessarily known, and may not display.

```
% /usr/etc/atmarp -al
<IP address>   port VPI VCI    flags
IP_addr         #    #   ###   0x82 CONN
 ATM addr: 0xATM_address
```

11. The IRIS ATM configuration information appears to be complete and correct. You are finished with this verification procedure.

2.4.2 Verifying the ATM Signaling Protocol Stack with `sigtest`

The `sigtest` utility verifies that the IRIS ATM signaling software is functional. Do these steps immediately after the initial installation, configuration, and startup. This test requires that the system be connected to an ATM switch. For complete information about the parameters used during this test, see Table 7, page 70.

1. Before starting this verification procedure, complete the verification procedure described in Section 2.4.1, page 60.
2. Open two shell windows and become super user (root).
3. Disable each ATM logical network interface that is serviced by the port you want to test, using the following command line:

   # **/usr/etc/ifconfig** *atm#* **down**

   The *atm#* variable is replaced with each enabled logical network interface.
4. In one window, make the ATM subsystem register with the switch for receiving SVC setup requests, using the following command lines to specify the traffic contract:

```
# /usr/lib/atm/bin/sigtest /hw/atm/port#
[0] Quit
[1] Register to accept incoming calls
[2] Attempt to setup a point-to-point call
[3] Attempt to setup a point-to-multipoint call
Enter choice: 1
Enter fwdMaxCSDU size: 9180
Enter bwdMaxCSDU size: 8192
Enter blliCode: 1
```

The registration is successful when the following message appears:

```
Registering for BLLI=1, fwd/bwdMaxCSDU=9180/8192
Registration successful.
```

Note: If the registration fails, look up the cause number in Table 29, page 203, or Table 30, page 206, or the error message in the alphabetical listing in Chapter 5, page 153.

5. In the other window, use the values shown in the following command lines to run a test that verifies the `atmsigd` command's ability to transmit and receive over a point-to-point SVC for best-effort traffic with the specified traffic contract. In the following example, values shown in *italics* are values that you need to create, values in **`bold`** should be entered exactly as illustrated, and text within <angle brackets> is displayed by `sigtest`.

```
# /usr/lib/atm/bin/sigtest   /hw/atm/port#
[0] Quit
[1] Register to accept incoming calls
[2] Attempt to setup a point-to-point call
[3] Attempt to setup a point-to-multipoint call
Enter choice: 2
Using calling address: address type = <type>
    Address =  <local ATM address is displayed>
Enter called address:
    Address type [1 for E164, or 2 for NSAP]: type_from_above
    Enter a string of exactly 40 hex characters:
          ATMaddress_copied_from_line_above
Enter fwdMaxCSDU size: 12280
Enter bwdMaxCSDU size: 2048
Enter number of BLLIs to offer [valid range 0-3]: 1
Enter blli 1: 3
Enter bearerClass: 3
Enter Forward Cell Rate
  Enter Cellrate type: 7
  Enter Peak Cell Rate for CLP 0+1: 357142
Enter Backward Cell Rate
  Enter Cellrate type: 7
  Enter Peak Cell Rate for CLP 0+1: 178571
Enter forwardQoS [0 - 4]: 0
Enter backwardQoS [0 - 4]: 0
```

6. Verify that the test succeeded by comparing your terminal display with the following examples, which illustrate the wording that appears in each window when the test is successful. In the transmitting window, you see the following lines:

```
Setup successful, negotiated fwd/bwdCSDU = 9180/2048
(1) Wrote 9180 bytes
(2) Wrote 9180 bytes
. . .
(9) Wrote 9180 bytes
(10) Wrote 9180 bytes
```

In the receiving window, you see these lines. Notice that the receiver's transmission cell rate matches the value you entered for the backward

(returning VC) cell rate. In contrast, the convergence sublayer data units (CSDU) values are expressed from the transmitter's point of view.

```
Listen successful:
  userHandle = 0x#, callHandle = #
  fwdMaxCSDU = 9180, bwdMaxCSDU = 2048
  BLLI = 3
  Xmit Cell Rate : cellrate spec. type = BEST EFFORT
  Peak Cell Rate for CLP 0+1 = 178571
  Caller Address : address type = NSAP
  Address = 47000580ffe1000000f115098d080069042aca00
Accepted connection!!
  Received 9180 bytes
  Received 9180 bytes
   . . .
   Received 9180 bytes
   Received 9180 bytes
Read 10 CSDUs for a total of 91800 bytes
```

7. In the receive window, again make the ATM subsystem register with the switch for receiving SVC setup requests, using the following command lines:

```
Enter choice: 1
Enter fwdMaxCSDU size: 12280
Enter bwdMaxCSDU size: 4096
Enter blliCode: 1
```

The registration is successful when this message appears:

```
Registering for BLLI=1, fwd/bwdMaxCSDU=12280/4096
Registration successful.
```

8. In the transmit window, run another test, using the values shown below, to verify the `atmsigd` command's ability to transmit and receive over a point-to-point VC for constant bit rate (CBR) traffic.

```
Enter choice: 2
Using calling address: address type = <type>
  Address =  <local ATM address is displayed>
Enter called address:
   Address type [1 for E164, or 2 for NSAP]: type_from_above
   Enter a string of exactly 40 hex characters:
     ATMaddress_copied_from_line_above
Enter fwdMaxCSDU size: 12280
Enter bwdMaxCSDU size: 512
Enter number of BLLIs to offer [valid range 0-3]: 0
Enter bearerClass:  4
Enter Forward Cell Rate
   Enter Cellrate type:  3
   Enter Peak Cell Rate for CLP 0+1: 14534
Enter Backward Cell Rate
   Enter Cellrate type:  3
   Enter Peak Cell Rate for CLP 0+1: 9259
Enter forwardQoS [0 - 4]:  1
Enter backwardQoS [0 - 4]: 1
```

Note: On a CHALLENGE or Onyx system, the cell rate values used in this test assume that the IRIS ATM board is configured with default settings. If the high-priority transmission rate queues have been configured with different settings, replace the cell rates used in the test with values that are currently configured, as reported by `atmstat -q`.

9. Verify that the test succeeded. When the test is successful, you see the following displays in the two windows. In the transmitting window:

```
Setup successful, negotiated fwd/bwdCSDU = 12280/512
(1) Wrote 12280 bytes
(2) Wrote 12280 bytes
. . .
(9) Wrote 12280 bytes
(10) Wrote 12280 bytes
```

In the receiving window:

```
Listen successful:
  userHandle = 0x#, callHandle = #
  fwdMaxCSDU = 12280, bwdMaxCSDU = 512
  BLLI = 0
  Xmit Cell Rate : cellrate spec. type = PEAK AGGREGATE
  Peak Cell Rate for CLP 0+1 = 9259
  Caller Address : address type = NSAP
  Address = 47000580ffe1000000f115098d080069042aca00
Accepted connection!!
  Received 12280 bytes
  Received 12280 bytes
  . . .
  Received 12280 bytes
  Received 12280 bytes
Read 10 CSDUs for a total of 122800 bytes
```

10. If you wish, you may perform additional verification by selecting other `sigtest` menu options and testing with other values. Table 7, page 70, summarizes the meanings and IRIS ATM support for the various parameters. Be aware that not all switches support the full range of options that this test makes available.

Table 7. Variables for the `sigtest` Utility

Item in `sigtest` prompt	Value	Description	Supported for use with `sigtest`
`maxCSDU`:		Maximum byte size for ATM AAL5 convergence sublayer data units, represented in decimal format. Supported range is 0 through 65,535.	Y
`BLLI`:		Broadband low-layer information for verification of protocol stack match at OSI layers 2 and 3 between endpoints.	
	0	Null.	Y
	1	Accept any. Only valid for receiving VC.	Y

Item in `sigtest` prompt	Value	Description	Supported for use with `sigtest`
	2	Level 2 LLC (SNAP).	N[1]
	3	LAN Emulation control.	Y
	4	LAN Emulation 802.3 data.	Y
	5	LAN Emulation 802.3 multicast.	Y
	6	LAN Emulation 802.5 data.	Y
	7	LAN Emulation 802.5 multicast.	Y
`Cellrate type`:			
	0	Null.	N
	1	Peak.	N
	2	Peak Tagged.	N
	3	Peak Aggregate.	Y
	4	PSB(peak cellrate, sustainable cellrate, maximum burst size).	N
	5	PSB Tagged.	N
	6	PSB Aggregate (peak cellrate, sustainable cellrate, maximum burst size).	Y
	7	Best Effort.	Y
`peak cellrate`:		Cells per second represented in decimal format for the peak transmission rate. For CBR, if the high-priority rate queues are configure with non-default values, the forward peak rate must exactly match a configured rate queue as displayed by `atmstat -q`.	Y
`sustainable cellrate`:		Cells per second represented in decimal format for the sustained (average) transmission rate. The value must be less than the peak cell rate defined for this VC. Software rounds up the entry to the next supported rate.	

1 Level 2 LLC requires that each CSDU (packet) contain a valid SNAP header. The IRIS ATM driver creates these when servicing IP traffic, but does not support this for `sigtest`.

Item in `sigtest` prompt	Value	Description	Supported for use with `sigtest`
`burst size`:		Cell count represented in decimal format for the maximum size of a transmission burst (that is, a tightly packed sequence of cells). Must be between 32 and 2048. Value entered is rounded up to a multiple of 32.	
`bearerClass`:			
	1	Bearer class A (BCOB-A).	Y
	2	Bearer class C (BCOB-C).	Y
	3	Bearer class X (BCOB-X) with unspecified, best-effort traffic type.	Y
	4	Bearer class X (BCOB-X) with CBR traffic type.	Y
	5	Bearer class X (BCOB-X) with VBR traffic type.	N
`QoS`:			
	0	Quality of service class 0; unspecified, best effort.	Y
	1	Quality of service class 1; Service Class A / CBR.	Y
	2	Quality of service class 2: Service Class B / VBR.	Y
	3	Quality of service class 3; Service Class C / Connection-oriented data.	Y
	4	Quality of service class 4: Service Class D / Connectionless data.	Y

2.4.3 Verifying IP over ATM Functionality

This section describes procedures for verifying that the IRIS ATM subsystem is servicing IP-based upper-layer applications. Separate instructions are provided for IP-over-PVC and IP-over-SVC. Follow both procedures if your system uses both PVCs and SVCs; otherwise, follow the set of procedures that is appropriate. At a minimum, the following configuration procedures must be performed before this verification procedure is tried:

- Change default settings for rate queues, Section 3.3.3.2, page 95.
- Map names to IP addresses, Section 3.5.2, page 106.
- Map IP addresses to network interfaces, Section 3.5.3, page 106.

- Map IP interfaces to the ATM subsystem, Section 3.5.5, page 109.
- For IP-over-SVC, configure the IRIS ATM signaling daemon, Section 3.6.1, page 123, and configure the ILMI daemon, Section 3.7.1, page 126.

Note: The procedures described in the following sections do not troubleshoot the IRIS ATM board. Hardware verification is explained in the *IRIS ATM\x15 OC3c Board for CHALLENGE and Onyx Installation Instructions* and in Section 2.4.4, page 75.

2.4.3.1 Verify IP-over-PVCs

Perform the following steps to verify that IP-over-PVC applications can transmit and receive through the IRIS ATM subsystem.

1. Perform the verification procedure described in Section 2.4.1, page 60.

   If `netstat` lists some IRIS ATM network interfaces that are not configured or are disabled, this is only a problem if you expect these to be functional. The following tests can be performed on all enabled interfaces. If an interface that you want to test is configured with an IP address but has an asterisk after it, use `ifconfig atm# up` to enable it.

2. Verify that the system is physically connected to another ATM endpoint (for example, an ATM switch or another ATM network controller).

3. From the `atmarp -a` display, select one of the PVC destination IP addresses for a different ATM station. The network portion of the destination IP addresses must match the network portion of one of the local IP addresses (as displayed by the *netaddress* entry from the `netstat` display). Use the selected destination address in the following two verification procedures.

4. Use `ping -r` to contact the ATM station, as follows:

```
% /usr/etc/ping -r IPaddress
PING name (IPaddress): 56 data bytes
64 bytes from IPaddress: icmp_seq=0 ttl=254 time=2 ms
64 bytes from IPaddress: icmp_seq=1 ttl=254 time=3 ms
64 bytes from IPaddress: icmp_seq=2 ttl=254 time=5 ms
64 bytes from IPaddress: icmp_seq=3 ttl=254 time=2 ms
CNTL-C
----name PING Statistics----
4 packets transmitted, 4 packets recvd, 0% packet loss
round-trip min/avg/max = 2/3/5 ms
```

5. Use `rcp` (or `rlogin`) to verify that the basic, IP-based network utilities are functional, as follows:

```
% /usr/bsd/rcp  guest@IPaddress:/path/filename  .
% /sbin/ls
<filename should be listed>
```

2.4.3.2 Verify IP-over-SVCs

Perform the following steps to verify that IP applications can transmit and receive through the IRIS ATM signaling subsystem.

1. Perform the verification procedure described in Section 2.4.1, page 60.

   If `netstat` lists some IRIS ATM network interfaces that are not configured or are disabled, this is only a problem if you expect these to be functional. The following tests can be performed on all enabled interfaces. If an interface that you want to test is configured with an IP address but has an asterisk after it, use `ifconfig`.

2. Verify that the system is physically connected to an ATM switch.

3. Obtain the IP address or name of at least one other ATM station on the same subnetwork as the physical port you want to test. One method for doing this step is to list the destinations known to this system (as shown in the following), then select one of the addresses displayed:

```
% grep netaddress /etc/hosts
```

   The *netaddress* variable specifies the nonmasked portion of the network address as displayed by the `netstat` command.

4. Use `ping -r` to contact the selected ATM station, as follows:

```
% /usr/etc/ping -r IPaddress
PING name (IPaddress): 56 data bytes
64 bytes from 150.166.76.26: icmp_seq=0 ttl=254 time=2 ms
64 bytes from 150.166.76.26: icmp_seq=1 ttl=254 time=3 ms
64 bytes from 150.166.76.26: icmp_seq=2 ttl=254 time=2 ms
64 bytes from 150.166.76.26: icmp_seq=3 ttl=254 time=2 ms

CNTL-C
----name PING Statistics----
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 2/2/3 ms
```

5. Use `rcp` (or `rlogin`) as follows to verify that the basic, IP-based network utilities are functional:

```
% /usr/bsd/rcp guest@IPaddress:/path/filename .
% /sbin/ls
<filename from other file system should be listed>
```

6. You can verify the creation of the SVC with the `atmarp -aln` command, which displays the address mappings for IP traffic. The single VC shown in the following example maps IP address 192.0.2.3 to ATM NSAP address 0x47.0005. The indicated VPI/VCI value was valid only for the first SVC to this address; subsequent SVCs to this address may use different VPI/VCI values.

```
% /usr/etc/atmarp -al
IP address   port VPI VCI flags
192.0.2.3     0    0  117 0x83 COMPL CONN
  ATM addr: 0x47.0005.80.ffe100.0000.f21a.024f.00204809006f.00
```

2.4.4 Verifying the IRIS ATM Hardware using `atmtest`

The `atmtest` utility verifies that the IRIS ATM port is functional. Perform this test when you suspect something is wrong with the hardware.

This test requires that the problematic port be physically connected to any of the following:

- To a switch. The local port's PVC to the switch (default address is VPI=0, VCI=201) must be mapped, at the switch, to an outgoing PVC that returns to the same local port.
- Directly to another IRIS ATM port, on the same or a different system. No switch exists between the two systems. The port must be reconfigured via the `atmconfig` command so that it recovers the clock from its own transmit clock signal:

  ```
  # atmconfig -i /hw/atm/port# -du
  # atmconfig -i /hw/atm/port# -o 0
  ```

  For more information about the use of the `-du` option, see Section 3.3.4.3, page 100.
- To itself with a low-loss loopback test connector (sold as an FDDI station testing device). The type of connector must be appropriate for the port you are testing (for example, MIC or dual-SC). Figure 26, page 77, shows an example of this type of device. The port must be reconfigured with the `atmconfig` command so that it generates the transmit clock from its own clock signal:

  ```
  # atmconfig -i /hw/atm/port# -du
  # atmconfig -i /hw/atm/port# -o 0
  ```
- On a CHALLENGE or Onyx system only, to itself with a loopback cable assembly that consists of a MIC-to-ST dual fiber optic cable with an adapter for connecting the two ST connectors to each other, as shown in Figure 27, page 77. The port must be reconfigured with the `atmconfig` command so that it generates the transmit clock from its own clock signal:

  ```
  # atmconfig -i /hw/atm/port# -du
  # atmconfig -i   /hw/atm/unit# -o 0
  ```

  Caution: Use the following command to reconfigure the port back to normal once the loopback cable is removed. Failure to do this causes the board to malfunction for normal switch use.

  ```
  # atmconfig -i /hw/atm/port# -o 1
  ```

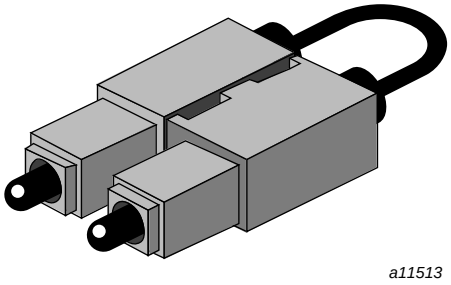


Figure 26. Loopback or Test Cable Assembly for Dual–SC Connector

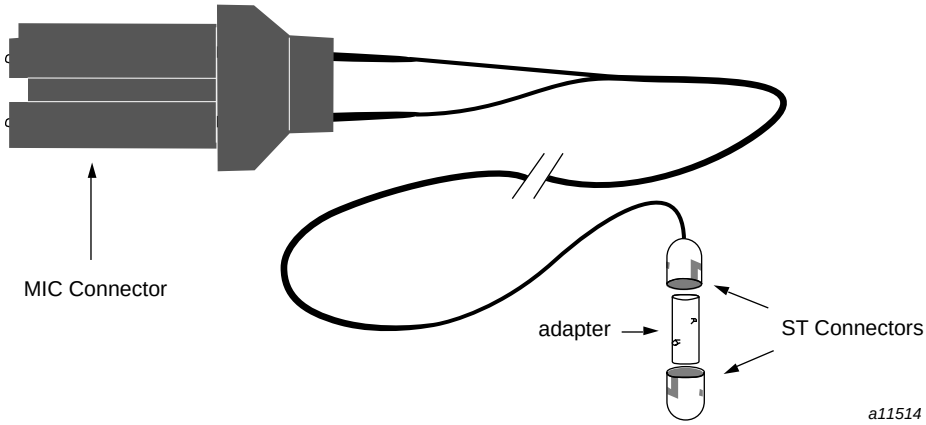


Figure 27. Loopback Cable Assembly and Adapter for MIC Connector

To verify that the IRIS ATM port is functional, follow these steps:

1. Make sure that the port is physically connected by one of the loopback methods explained previously.
2. As super user, use the following command line to invoke `atmtest` to transmit and receive through one IRIS ATM port using default settings for VPI/VCI and rate. Invoke this command once for each port.

   # /usr/etc/atmtest -i /hw/atm/*port#* **-Xrw &**

The *port#* variable indicates the port's assigned number or, on a CHALLENGE or Onyx system, the board's unit number.

The resulting display should indicate success with wording similar to the following example. You can terminate the test at any time by pressing `Ctrl-C`.

```
atmtest: /hw/atm/port#:
vpi/vci = 0 201 xmit-rate: 68.57 Mbps
- 1000/10000 frames transmitted, total 0 lost
. . .
--- 10000 frames transmitted, 0 lost ---
```

3. Invoke `atmtest` to test a number of different transmission rates, using the following command lines. On a CHALLENGE or Onyx system, you can test each of the transmission rates configured on the board's rate queues; use the `atmstat` command to display the rates. (See Section 1.7.2.1, page 46, for detailed information on rate queues.) On an Origin or Onyx2 system, you may test any rate you want. With each invocation of `atmtest`, you specify a different rate in bits per second. As the test precedes, it displays its progress. You can kill each test, at any time, by simultaneously pressing the `Ctrl` and `c` keys. Use this format:

   # /usr/etc/atmtest -i /hw/atm/*port#* **-e***bitspersecond* **&**

   Each entry has the following meaning:

   - The *port#* variable specifies a port number displayed by `/sbin/hinv` for an IRIS ATM board. On a CHALLENGE or Onyx system, use the board's unit number as the port number.
   - The *bitspersecond* variable specifies a decimal number indicating the speed in user payload bits per second at which the data is to be transmitted. On a CHALLENGE or Onyx system, this must match one of the rates supported by the board; use the `atmstat -i port# -q` command to list the currently configured rates.

   For example:

```
# /usr/etc/atmtest -i /hw/atm/0 -Xrw -e10000000
atmtest: /hw/atm/0:
vpi/vci = 0 201 xmit-rate: 10.00 Mbps Best Eff
- 1000/10000 frames transmitted, total 0 lost
...
# /usr/etc/atmtest -i /hw/atm/port# -Xrw -e30000000
# /usr/etc/atmtest -i /hw/atm/port# -Xrw -e68000000
# /usr/etc/atmtest -i /hw/atm/port# -Xrw -e135991460
```

4. When you are finished testing and ready to reattach the system to its switch, use this command to reconfigure the port back to normal:

```
# atmconfig -i /hw/atm/port# -o 1
```

IRIS ATM Configuration Reference [3]

This chapter provides a complete explanation for every configuration procedure available for the IRIS ATM product.

IRIS ATM can be configured to support any or all of the functionalities listed below. Before you perform any configuration procedures, you should know which of these options this system needs to support:

- IP-over-SVCs: IP applications over switched virtual circuits (SVCs) in compliance with RFC 1577
- Non-IP-over-SVCs: non-IP applications (developed by the customer) over SVCs
- IP-over-PVCs: IP applications over permanent virtual circuits (PVCs)
- Non-IP-over-PVCs: non-IP applications (developed by the customer) over PVCs

3.1 Complete List of Configurable Parameters

Table 8, page 82, lists all of the IRIS ATM parameters that are configurable. Configuration for many of these items is optional, since the IRIS ATM software contains default settings. The table shows which parameters are required (R) and which are optional (O). The table also indicates the location for the configuration instructions. Some items apply only to switched virtual circuits (SVCs) or to permanent virtual circuits (PVCs). In most reconfigurations, once the configuration task is complete, you need to restart one or more sections of the IRIS ATM subsystem in order to start using the new configuration. Table 19, page 132, summarizes methods for smoothly stopping and starting the different parts of the IRIS ATM subsystem.

Table 8. Complete List of Configurable Parameters

Parameter Category	Parameter to configure	Required / Optional	Section where instructions are located
IRIS ATM HIO board for CHALLENGE and Onyx systems			
	Rate queues	O	Section 3.3.3.2, page 95
	On-the-fly changes to rate queue settings	O	Section 3.3.4, page 98
	Board unit numbering	O	Section 3.3.2, page 93[1]
	TCP/UDP checksumming done by hardware	O	Section 3.4, page 102
	Method for recovering the SONET clock	R only when not using a switch	Section 3.3.4.4, page 100
	Number of active VCs supported	O	`atmconfig` man page (see section on increasing buffers)
IRIS ATM XIO board for Origin2000 and Onyx2 systems			
	TCP/UDP checksumming done by hardware	O	Section 3.4, page 102
	Method for recovering the SONET clock	R only when not using a switch	Section 3.2.2.3, page 87, or Section 3.2.3, page 88
	Number of active VCs supported per port	O	`atmconfig` man page (see section on increasing buffers)
	Port parameters	O	Section 3.2.3, page 88

1 Equivalent instructions for setting the unit number by installing jumpers on the board are provided in the *IRIS ATM\x15 OC3c Board for CHALLENGE and Onyx Installation Instructions*. Either method can be used.

Parameter Category	Parameter to configure	Required / Optional	Section where instructions are located
IRIS IP-over-ATM driver			
	Size of maximum transmission unit and AAL5 protocol data unit	O	Section 3.4.2, page 102
	Remove support for IP from driver	O	Section 3.4.1, page 102
	Number of IP-over-ATM network interfaces (`atm#`) created at boot time[2]	O	Section 3.4, page 102
IP Network Interfaces over ATM (LIS)			
	IP addresses and names	R	Section 3.5.2, page 106
	Network interface to IP–address mappings	R	Section 3.5.3, page 106
	ATM-to-IP address mappings, including designation of ATMARP server (address resolution) and port usage	R	Section 3.5.5, page 109
	Transmission rate for VCs carrying IP traffic	O	Section 3.5.7.1, page 119
	Optional IP parameters, such as subnetwork mask	O	Section 3.5.4, page 107
	LLC/SNAP encapsulation	O	Section 3.5.5.1, page 109
	RFC1577 LIS Configuration	R	Section 3.5.5.2, page 111 Section 3.5.7, page 118
	For SVCs only: on-the-fly changes to SVCs and PVCs used by IRIS ATM signaling and ILMI	O	Section 3.5.7, page 118
IRIS ATM signaling (SVCs)			

2 Unlike many LAN protocols, ATM can be configured to support multiple logical network interfaces for each physical port. For example, two or more IP addresses can be used for a single IRIS ATM-OC3c port.

Parameter Category	Parameter to configure	Required / Optional	Section where instructions are located
	ATM UNI and signaling	R	Section 3.6.1, page 123
	VPI/VCI used for signaling	O	Section 3.6.2, page 124
IRIS interim local management interface (ILMI)			
	ILMI module	R	Section 3.7.1, page 126
	Runtime options (UDP socket number, error message level) for the ILMI daemon	O	Section 3.7.2, page 128
	Local port's ATM address	O	Section 3.7.1, page 126
	Management information base (MIB)	O	Section 3.7.4, page 129

3.2 IRIS ATM-OC3c 4Port XIO Board Configuration

This section provides instructions for configuring the four-port IRIS ATM XIO board for the Origin2000 and Onyx2 platforms. None of the procedures in this section are required in most configurations. Two software methods are available for controlling and configuring the board:

- Edit files that configure the board each time it is powered up or reset.
- Use the command line utility (`atmconfig`) that changes the parameter immediately.

3.2.1 Assigning Board Unit and Port Numbers for XIO Hardware

The `ioconfig` utility assigns unit numbers to IRIS ATM XIO boards. For complete details, see the `ioconfig`(1M) man page.

Port numbers are assigned to the ATM XIO ports by the following mathematical formula:

First port (labeled `port 0` on panel plate) = *board_unit*# * 4

Second port = *first_port*# + 1

Third port = *first_port*# + 2

Fourth port = *first_port*# + 3

For example, a board that is assigned unit 0 has ports 0 through 3 while a board assigned unit 2 has ports 8 through 11.

3.2.1.1 Displaying Unit and Port Assignment Information

You can use `hinv`, as shown in the following display, to verify or discover the board unit and port number assignments:

```
% /sbin/hinv -mvv | grep ATM
QUAD_ATM Board:  barcode ######  part 030-0948-00# rev #
ATM XIO 4 port OC-3c: module module#, slot slot#, unit unit# (ports: #-#)
```

Each installed board should have two lines, similar to those above. *unit*# indicates the unit that was assigned to the board in the indicated location (*module*# and *slot*#).

If any board is not listed by the `hinv` display, contact the person responsible for your site's hardware installation. One or more of the following problems may be the reason a board is not listed by `hinv`:

- The board may be improperly installed (for example, it may be loose or its IO slot may not be activated).
- The IRIS ATM board may be dysfunctional.

3.2.2 The `atmconfig` Utility

The `atmconfig` utility, which is used to dynamically change configuration settings on the IRIS ATM hardware, is provided with the IRIS ATM software. The configurations take effect immediately. With the next reboot, the settings return to the default settings. The `atmconfig` man page provides complete usage details. This section provides an overview and detailed instructions for common tasks.

The `atmconfig` command can be used to perform the following port configuration tasks:

- Display port's operational configuration parameters and currently loaded firmware version.

- Bring a port up or down, without resetting it. When in the down state, the port does not transmit or receive data over its physical (SONET) connections, but it can communicate with the host and driver.
- Reset and reinitialize an ATM port. Any in-progress data is lost. During reinitialization, the utility establishes communication between the IRIS ATM hardware and the driver within the operating system.
- Configure a port's source for its SONET transmit clock. This configuration step is required when a port is attached directly to an endpoint (without an intermediate switch) or when a loopback cable is attached to a port.

The `atmconfig` command configures the hardware, not the logical (software) network interface. To make changes on port 1, use the string `-i/hw/atm/1` as an argument to the command line. To select port 2, use the string `-i/hw/atm/2`. When the `-i/hw/atm#` argument is not supplied, the action is done to port 0 (that is, `/hw/atm/0`).

Note: The `atmconfig` utility also provides status information. See Section 4.1, page 135, for these features.

3.2.2.1 Resetting a Port

To reset a port (that is, put it into the pre-initialized (`PRE-INIT`) state), use a command line like the following one shown. Each reset affects a pair of ports: either the first pair on the board (labeled `port0` and `port1` on the panel plate) or the second pair (labeled `port2` and `port3`).

```
# /usr/etc/atmconfig -i# -r
```

The # option indicates the port's number (for example, `-i4` for port 4, resets devices `/hw/atm/4` and `/hw/atm/5`).

Caution: Any in-progress data is lost for both ports. Follow the instructions in Table 19, page 132, to ensure that other modules are smoothly stopped and restarted.

3.2.2.2 Changing the State of a Port

To reinitialize a port (that is, put it into the down state), or bring the port to its operational state (that is, put it into the up state), you must first reset the port to its pre-initialized state (see Section 3.2.2.1, page 86) and then use the appropriate command line from the two shown in in the following display:

```
# /usr/etc/atmconfig -i# -d
# /usr/etc/atmconfig -i# -u
```

The `-d` is for the down state and `-u` is for the up state, and # indicates the port number (for example, `-i0` for port 0, device `/hw/atm/0`, or `-i1` for port 1).

The `-r` option resets pairs of ports while the `-d` or `-u` option changes the state of a single port. Therefore, you must explicitly bring each port into its up or down state. For example, if you want to bring port 3 down and then up, you must reset, then bring down and bring up ports 2 and 3, as follows:

```
atmconfig -i2 -r
atmconfig -i2 -d
atmconfig -i3 -d
atmconfig -i2 -u
atmconfig -i3 -u
```

On Challenge and Onyx systems, the `-r`, `-d`, and `-u` options work in a different way. For information on these options on Challenge and Onyx systems, see Section 3.3.4.3, page 100.

3.2.2.3 Enabling the ATM Port to Function in a Configuration without a Switch

For an IRIS ATM port to function when physically looped back to itself (the output line feeds into the same port's input line) or when connected to other ATM systems that are not switches, the port must be configured to use its own clock as the source for its SONET transmit clock. By contrast, when connected to a switch, the port uses the default clock, which is on the incoming line. To operate without a switch for a short period of time, use the following instructions. For long-term operation without a switch, use the instructions in Section 3.2.3, page 88.

For a temporary configuration change, use the following command line:

```
# /usr/etc/atmconfig -i# -o 0
```

The # option indicates the port number (for example, `0` for port 0, or `1` for port 1).

> **Note:** Before reconfiguring the port, disable the ATM daemons. The signaling and management daemons (`atmsigd` and `atmilmid`) must not be running when there is no switch available.

To change the clock source back to the default, use the following command line. This setting is appropriate when a port is connected to an ATM switch.

```
# /usr/etc/atmconfig -i# -o 1
```

The # option indicates the port number (for example, `0` for port 0, or `1` for port 1).

3.2.3 The `atmhw.conf` File for XIO Board

The `/var/atm/atmhw.conf-#` file configures operational features of each port on the IRIS ATM-OC3c 4Port XIO board so that the changes survive from one reboot to the next. Changes made in this file take effect with the next initialization of the hardware (for example, reboot of the system, restart of the IRIS ATM driver, invocation of the `atmconfig -F` *filename* command, or reset of the port).

Use the following steps to configure a port that it is going to be used without an ATM switch over an extended length of time:

1. Open the `/var/atm/atmhw.conf-#` file for the port. If this file does not exist, make a copy of the standard `/var/atm/atmhw.conf` file and give the new file a name that includes the port's assigned number. For example, for a port that has been assigned a value of 8, name the new file `/var/atm/atmhw.conf-8`. Remove the leading pound sign (#) from the `PHYOPTS` entry. The line should look like the following example:

   ```
   PHYOPTS 0
   ```

2. Verify that there are no entries for this port in the LIS configuration files (for example, the `/var/atm/pvc.conf` and `/var/atm/ifatm.conf` files).

Note: Additional information about port hardware configuration options that are available in this file are documented in the `atmconfig`(1M) man page.

3.3 IRIS ATM-OC3c Mezzanine HIO Board Configuration

This section provides instructions for configuring the IRIS ATM board for the CHALLENGE and Onyx platforms. In this section, the only required step is changing the runtime rates on transmission queues, described in Section 3.3.3.2, page 95, and this step is needed only when one or more of your constant bit rate (CBR) channels require a non-default rate. Table 12, page 95, lists the default rates.

Table 9, page 90, lists all of the parameters on the IRIS ATM-OC3c board that can be controlled. Two software methods are available for controlling and

configuring the board, as described in the following section and summarized in Table 9, page 90:

- Edit files that configure the board each time it is powered up or reset.
- Use the command line utility (`atmconfig`) that changes the parameter immediately.

Table 9. Configurable Parameters for the IRIS ATM-OC3c HIO Board

Board parameter	Runtime configuration	On the fly configuration
Set manner in which unit numbers are assigned to all IRIS ATM boards during startup.	Edit `/var/sysgen/master.d/atm` file, explained in Section 3.3.2, page 93.	Not available
Fix the transmission rate setting on one or more rate queues so that the configured rates are always available.	Edit `/var/atm/atmhw.conf` file, explained in Section 3.3.3.2, page 95.	Use `atmconfig` command, as explained in Section 3.3.3.2, page 95.
Change operational status	Not available	Use `atmconfig` command, as explained in Section 3.3.4.3, page 100.
Reset board	Not available	Use `atmconfig` command, as explained in Section 3.3.4.2, page 99.

3.3.1 Assigning Board Unit Numbers

Unit numbers are assigned to IRIS ATM boards by one of the following methods: software dynamically assigns the unit numbers based on the order the boards are discovered during power up; or the jumpers assign the unit numbers. The default method is the dynamic software method, however, assignment from the jumpers can be configured by editing the board configuration file as described in Section 3.3.2, page 93. The advantages and disadvantages of each of these methods is described separately in Table 10, page 91, and Table 11, page 91.

Table 10. Jumper-assigned Unit Numbers: Advantages and Disadvantages

Advantages	Disadvantages
The unit number for any particular board is always the same, from one power up to the next.	The numbering for IRIS ATM boards may not be sequential. For example, there may be a unit 2 but no unit 0 or 1.
Regardless of what happens to other boards (new installations, removals, or dysfunctions), an IRIS ATM board's unit number reflects its jumper settings.	
A script that uses `ifconfig`, `atmconfig`, `atmstat`, or `atmtest` always performs its operations on the same physical boards.	The unit numbering does not reflect the ordering of the boards within the card cage. For example, unit 1 may be located on the main IO4 board while unit 0 is located on the fourth IO4 board in the system.

Table 11. Software-assigned Unit Numbers: Advantages and Disadvantages

Advantages	Disadvantages
The IRIS ATM boards are always numbered sequentially, starting at 0 (that is, unit 0, unit 1, etc.).	The unit number for any particular board can vary from one power up to the next. If a different set of upstream IRIS ATM boards are located during startup, a board's unit number changes. The following conditions can cause a unit number to change: • an upstream board is removed • an upstream board is dysfunctional • a new board is installed upstream
The unit and interface numbering always reflect the ordering of the boards within the chassis. For example, unit 0 is always located upstream from unit 1.	A script that uses `ifconfig`, `atmconfig`, `atmstat`, or `atmtest` may perform its operations on different physical boards after a system restarts.

3.3.1.1 Displaying Unit Assignment Information

You can use `hinv`, as shown in the following line, to verify or discover the unit assignments:

```
% /sbin/hinv
...
ATM OC-3 unit unit#: slot slot#, adapter adapter#
```

Each installed board should have one line, similar to the preceding line. *unit*# indicates the number that is assigned to the board installed in the indicated slot (*slot*#) and mezzanine adapter position (*adapter*#, where 5 indicates lower mezzanine position and 6 indicates upper).

If any board is not listed by the `hinv` display, contact the person responsible for your site's hardware installation. One or more of the following problems may be the reason a board is not listed by `hinv`:

- The jumper setting on the missing board may be a duplicate of one that is displayed.
- The board may be improperly installed (for example, it may be loose or its IO4 board may be loose).
- The IRIS ATM board or the IO4 board onto which it is attached may be dysfunctional.
- If no IRIS ATM boards are listed, the version of IRIX that is currently running may not support IRIS ATM or the IRIS ATM software may not be installed.

3.3.1.2 The Software (Dynamic) Assignment Method

When the IRIS ATM software (driver) dynamically assigns unit numbers to IRIS ATM-OC3c HIO boards, it assigns unit 0 to the first IRIS ATM port (board) that it locates, unit 1 to the second, unit 2 to the third, and so on. The following list shows the order in which the system searches for the IRIS ATM ports:

1. On the main IO4 board; the lower mezzanine adapter position (#5) and then the upper mezzanine position (#6).
2. On the next installed IO4 board; the lower mezzanine adapter position and then the upper position.
3. And so on, until there are no more IO4 boards installed.

The `hinv` display for a CHALLENGE L Deskside that has 3 IRIS ATM boards installed might appear as shown in the following example:

```
...
ATM OC-3c unit 0: slot 5, adapter 5
ATM OC-3c unit 1: slot 4, adapter 5
ATM OC-3c unit 2: slot 4, adapter 6
```

3.3.2 Configuring Board Unit Numbers

The `/var/sysgen/master.d/atm` file allows you to configure the IRIS ATM hardware driver so that it ignores or reads the hardware settings of the unit jumper set on all the IRIS ATM HIO boards in the system. In the default mode, which ignores hardware jumpers, the software dynamically assigns unit numbers, based on the order in which the boards are located during power up.

To configure the method used for assigning unit numbers, use the following steps:

1. Open the `/var/sysgen/master.d/atm` file and edit the following line:

   ```
   int atm_ignore_unit = #;
   ```

   To dynamically assign unit numbers by the software, the line should look like the following:

   ```
   int atm_ignore_unit = 1;
   ```

   To base the unit numbers on the jumper settings of each board, the line should look like the following:

   ```
   int atm_ignore_unit = 0;
   ```

   Note: The *IRIS ATM\x15 OC3c Board for CHALLENGE and Onyx Installation Instructions* manual describes how to set the unit jumpers. Changing the jumpers can be done only by Silicon Graphics trained personnel.

2. If this is the only, or the last configuration task, reboot the system. Otherwise, perform the other configuration tasks, then reboot.

3.3.3 ATM-OC3c HIO Board Transmission Rate Configuration

During startup, the IRIS ATM driver configures the ATM-OC3c Mezzanine HIO boards with the settings from one or more `/var/atm/atmhw.conf` configuration files. You can edit these files to change the default settings. Currently, the only configurable items are the transmission rates for the eight rate queues on each IRIS ATM board. (See Section 1.7.2.1, page 46, for a

description of these transmission rate queues and how they are managed by the IRIS ATM driver.)

3.3.3.1 Default Rates for Transmission Queues

The `atmhw.conf` file is shipped in the following format, where the high-priority rate queues are commented out (not configured by the file), thus leaving them open for configuration by the IRIS ATM driver as needed. The resulting default transmission rates are summarized in Table 12, page 95. In this file, the four high-priority queues are identified as a0 to a3; the high-priority queues are intended for constant bit rate (CBR) and variable bit rate (VBR) traffic, but can be used for best-effort traffic when there is no CBR traffic. The four low-priority queues are b0 to b3; these queues are intended for best-effort traffic, such as traditional IP applications and local area network traffic. The transmission rates in the configuration file are specified in bits of user (above ATM-layer payload) data per second.

```
#!/usr/etc/atmconfig -F
#
# atmhw.conf#
# This file contains values used by atmconfig(1m)
# to initialize the ATM OC-3c ‘‘rate queues’’
# during system boot.
#
# High-priority rate queues. Uncomment these
# lines to set the rate to a fixed setting.
# RATEQ a0 405405
# RATEQ a1 394736
# RATEQ a2 3555555
# RATEQ a3 5581395

# Low-priority rate queues.
RATEQ b0 10000000
RATEQ b1 30000000
RATEQ b2 68000000
RATEQ b3 135991460
```

Table 12. Default Transmission Rates on ATM-OC3c Queues

Rate queue Number ID	String ID	Default cell rate (in ATM cells per second)	Default bit rate (in user payload bits per second)	Priority / Use
0	a0	unconfigured	0	High / CBR, VBR[3]
1	a1	unconfigured	0	High / CBR, VBR
2	a2	unconfigured	0	High / CBR, VBR
3	a3	unconfigured	0	High / CBR, VBR
4	b0	26041	10000000	Low / BE
5	b1	78125	30000000	Low / BE
6	b2	178571	68000000	Low / BE
7	b3	357142	135991460	Low / BE

3.3.3.2 Changing the Runtime Rates on Transmission Queues

To change the default settings for one or more rate queues on a board, use the following instructions:

1. From Table 13, page 95, determine which file you need to edit. Note that to configure the rate queues for a specific board, you create (or, if it exists, you edit) one of the `atmhw.conf-#` files. Whereas, to configure queues for all the boards that do not have a specific `atmhw.conf-#` file, you edit the `atmhw.conf` file.

Table 13. Files for Configuring Rate Queues

Board	File to be edited
All installed boards that do not have their own individual `atmhw.conf-#` files.	`/var/atm/atmhw.conf`
For unit 0 only	`/var/atm/atmhw.conf-0`
For unit 1 only	`/var/atm/atmhw.conf-1`

[3] CBR = constant bit rate; VBR = variable bit rate; BE = best effort

Board	File to be edited
For unit 2 only	`/var/atm/atmhw.conf-2`
For unit *X* only	`/var/atm/atmhw.conf-X`

2. If the file that you are editing is `/var/atm/atmhw.conf`, go to the next step. Otherwise, check to see if the file you want already exists. If it does, go to the next step. If it does not, make a copy of `/var/atm/atmhw.conf`. Name the new file the name you identified from Table 13, page 95. Use a command like the following:

   ```
   % cp /var/atm/atmhw.conf /var/atm/atmhw.conf-#
   ```

   The # option is the board's unit number.

3. Open the file. Locate the line(s) for the rate queue(s) that you want to reconfigure. Figure 28, page 97, shows the format for the entries in this file.

4. If necessary, uncomment the line by removing the pound sign (#) from the front.

5. Edit the third field in the line. Figure 28, page 97, shows the format for the entries in this file. Appendix B, page 211, lists the rates that are supported. The value in this field is a decimal numeral within the range 0 to 137,142,800 indicating the number of bits of ATM payload data that is transmitted per second. Since each ATM cell carries 384 payload bits, you can convert a cellrate to payload bits per second by multiplying the desired cellrate times 384.

   If the value entered in this field does not match one of the supported rates (from Appendix B, page 211), then at startup, the board will be configured with the next higher supported rate.

   Note: The values that you specify for the high-priority queues are the peak rates that must be used by all the transmitting constant and variable bit rate VCs.

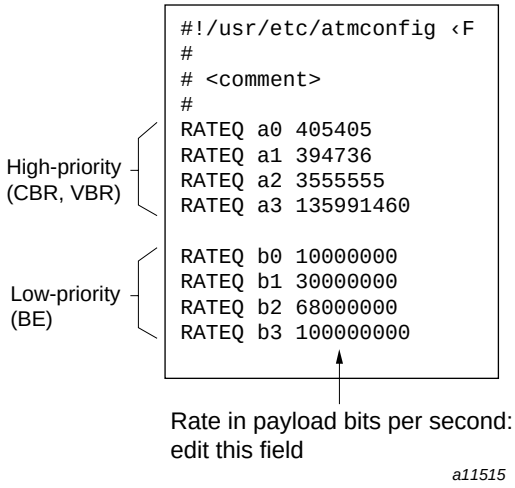


Figure 28. Format for `atmhw.conf` Files

Note: Whenever the `atmilmid` and `atmsigd` modules have overhead data to transmit, they need to use a very small, best-effort portion of the bandwidth (which can be up to 20,000 cells per second). In addition, the IRIS ATM default implementation of IP-over-ATM uses only the low-priority queues. So, do not set all the low-priority queues to zero if the system is using SVCs, the IRIS ATM ILMI daemon (`atmilmid`), or IRIS ATM's IP-over-ATM (with SVCs or PVCs).

6. For each line, verify that you have not entered any characters following the final digit of the rate value. For example, comments preceded by a pound sign (#) must be placed on a line of their own, not on the same line as a configuration entry.

7. If this is the only configuration task, use the following command lines to configure the board with your changes. Otherwise, perform the other configuration tasks, then restart the board (for restarting instructions, see Section 3.9, page 131).

```
# ifconfig atm#   down
<do this for each interface associated with this board>
# atmconfig -i# -r
# atmconfig -i# -d
# atmconfig -i# -F filename
# atmconfig -i# -u
# killall -Hup atmilmid
```

Note: When you finish, it is valid to have one `atmhw.conf` file to configure all boards except for those for which you created `atmhw.conf-#` files. For example, on a system with four IRIS ATM-OC3c boards, there could exist two files: `atmhw.conf` for units 0, 1, and 2; and `atmhw.conf-3` for board unit 3. It is also valid to not have an `atmhw.conf` file and an `atmhw.conf-#` file for each installed board.

3.3.4 The `atmconfig` Utility

The `atmconfig` utility is provided with the IRIS ATM software for dynamically changing configuration settings on IRIS ATM boards. The configurations take effect immediately. With the next reboot, the settings return to the default settings. The `atmconfig`(1M) man page provides complete usage details. This section provides an overview and examples of command lines.

Note: For information about configuring the unit number on an ATM-OC3c Mezzanine board, see the *IRIS ATM\x15 OC3c Board for CHALLENGE and Onyx Installation Instructions* or Section 3.3.2, page 93.

The `atmconfig` command can be used to perform the following board configuration tasks:

- Configure the rates for the IRIS ATM board's rate queues (timers).
- Display the board's operational configuration parameters and currently loaded firmware version.
- Bring a board up or down, without resetting it. When in the down state, the board does not transmit or receive data over its physical (SONET) connections, but it can communicate with the host and driver.
- Reset and reinitialize a board. Any in-progress data is lost. During reinitialization, the utility establishes communication between an IRIS ATM board and the driver within the operating system.
- Configure the source that the port uses for its SONET transmit clock. This procedure is required when a port is attached directly to an endpoint

(without an intermediate switch) or when a loopback cable is attached to the port.

The `atmconfig` command configures the hardware, not the logical (software) network interface. To make changes on board unit 1, use the string `-i1` as an argument to the command line. To select board unit 2, use the string `-i2`. When the `-i#` argument is not supplied, the action is done to unit 0.

Note: The `atmconfig` utility also provides status information. See Section 4.1, page 135, for these features.

3.3.4.1 Dynamically Changing Rates on Transmission Queues

To change the transmission rate for a rate queue, without restarting the system, use the following command line. If the queue is being used, this command will fail:

`/usr/etc/atmconfig -i` # **`-Q`** *queueID*`,`*payload_bits_per_second*

The # option indicates the hardware unit number (for example, `0` for board unit 0, device `/hw/atm/0`, or `1` for board unit 1, device `/hw/atm/1`), *queueID* indicates the queue (for example, `0`, `1`, `2`, or `3` for the high-priority queues and `4`, `5`, `6`, or `7` for the low-priority queues), and *payload_bits_per_second* indicates the rate in bits of user (above ATM-layer) data per second.

Note: There are 384 bits of user payload in each ATM cell.

For example, to set the first high-priority queue on board unit 0 to 2,307,840 payload bits per second (6,010 ATM cells per second), use this command:

```
# /usr/etc/atmconfig -i0 -Q 0,2307840
```

To set the second high-priority queue on board unit 3 to 96,000,000 payload bits per second (250,000 ATM cells per second), use this command:

```
# /usr/etc/atmconfig -i3 -Q 1,96000000
```

Note: Whenever the `atmilmid` and `atmsigd` modules have ATM-overhead data to transmit, they need to use a very small, best-effort portion (up to one-tenth) of the bandwidth configured on rate queue seven (priority b3), so do not set this queue to zero.

3.3.4.2 Resetting a Board

To reset a board (that is, put it into the pre-initialized state), use a command line like the following:

```
# /usr/etc/atmconfig -i# -r
```

The # option indicates the hardware unit number (for example, `0` for board unit 0, device `/hw/atm/0`, or `1` for board unit 1, device `/hw/atm/1`).

Caution: Any in-progress data is lost. Follow the instructions in Table 19, page 132, to ensure that other modules are stopped and restarted.

3.3.4.3 Changing the State of a Board

To put a board into the down state, or bring the board to the up state, use the appropriate command line from one of the following:

```
# /usr/etc/atmconfig -i# -d
# /usr/etc/atmconfig -i# -u
```

The `-d` is for the down state and `-u` is for the up state, and # indicates the hardware unit number (for example, `0` for board unit 0, device `/hw/atm/0`, or `1` for board unit 1, device `/hw/atm/1`).

On Challenge systems, when you use the `-d` option, the `-r` option is implicitly invoked (for more information about the `-r` option, see Section 3.3.4.2, page 99). In the following example, board unit 1 is reset, brought down, and brought up.

```
atmconfig -i1 -du
```

The previous example is the equivalent of the following commands:

```
atmconfig -i1 -r
atmconfig -i1 -d
atmconfig -i1 -u
```

On Origin2000 or Onyx 2 systems, the `-r`, `-d`, and `-u` options work in a different way. For information on these options on Origin2000 or Onyx 2 systems, see Section 3.2.2.2, page 86.

3.3.4.4 Enabling an ATM Port to Function in a Configuration without a Switch

For the ATM-OC3c board to function when it is physically looped back to itself (output line feeds into the same port's input line), or when it is connected to an ATM system that is not a switch, the board must be configured to use its own clock as its source for generating the SONET transmit clock, instead of using the clock on the incoming line, which is the default. To do this, use the following command line:

```
# /usr/etc/atmconfig -i# -o 0
```

The # option indicates the hardware unit number (for example, `0` for board unit 0, device `/hw/atm/0`, or `1` for board unit 1, device `/hw/atm/1`).

Note: The signaling and management daemons (`atmsigd` and `atmilmid`) must not be running when there is no switch available for them to interact with.

To change the clock source back to the default, use the following command line. This setting is appropriate when a port is connected to an ATM switch.

```
# /usr/etc/atmconfig -i # -o 1
```

The # indicates the hardware unit number (for example, `0` for board unit 0, device `/hw/atm/0`, or `1` for board unit 1, device `/hw/atm/1`).

3.3.5 The `atmhw.conf` File for HIO Board

The `/var/atm/atmhw.conf-#` file configures operational features for each IRIS ATM-OC3c HIO board so that the changes survive from one reboot to the next. Changes made in this file take effect with the next initialization of the hardware (for example, reboot of the system, restart of the IRIS ATM driver, invocation of the `atmconfig -F` *filename* command, or reset of the port).

Follow these steps to configure a port that it is going to be used without an ATM switch over an extended length of time:

1. Open the `/var/atm/atmhw.conf-#` file for the board. If this file does not exist, make a copy of the standard `/var/atm/atmhw.conf` file and give the new file a name that includes the board's assigned number. For example, for a board that has been assigned the unit 3, name the new file `/var/atm/atmhw.conf-3`. Remove the leading pound sign (#) from the `PHYOPTS` entry. The line should look like the following example:

   ```
   PHYOPTS 0
   ```

2. Verify that there are no entries for this port in the LIS configuration files (for example, the `/var/atm/pvc.conf` and `/var/atm/ifatm.conf` files).

Note: Additional information about hardware configuration options that are available in this file are documented in the `atmconfig`(1M) man page.

3.4 IRIS ATM IP Driver Configuration

This section describes how to configure the portion of the IRIS ATM driver that handles support for the IP protocol.

3.4.1 Building IRIS ATM without IP Support

IRIS ATM includes support for TCP/IP (that is, the IP protocol suite) by default. If you do not plan to ever use IP-over-ATM, use the following steps to configure the IRIS ATM driver for use as a non-IP connection.

1. Open the `/var/sysgen/system/atm.sm` file (for a CHALLENGE or Onyx system) or `quadoc3.sm` file (for an Origin or Onyx2 system)
2. Change the `INCLUDE` entry to `EXCLUDE` as shown here:

   The entry will look like the following example:

   ```
   INCLUDE: if_atm
   ```

   The entry should be changed to look like the following example:

   ```
   EXCLUDE: if_atm
   ```

   Note: Do not remove this line from the file. The driver cannot build correctly without the presence of either an `INCLUDE` or an `EXCLUDE` entry for `if_atm`.

3. Rebuild (`autoconfig`) the operating system to include a rebuilt IRIS ATM driver.

   Note: If you exclude IP support from the driver, and later you want to use IP, you must redo the above steps. If you configure the software to add IP functionality later, you need to do only the additional configuration steps. When the driver is built to support IP, but IP is not configured, some error messages are displayed each time the system is started.

3.4.2 Configuring IP Support in the IRIS ATM Driver

This section provides instructions for configuring operational parameters for the portion of the IRIS ATM driver that manages TCP/IP traffic. Table 14, page 103, lists the driver parameters that can be altered and indicates the default setting.

Table 14. IRIS ATM Driver Parameters

Item	Default setting	Description
`ifatm_mtu`	0 =9,180 bytes[4]	Maximum size for IP transmission unit datagrams. Valid values are 0 and 8 through 9,180 (decimal) inclusive. Setting this item to 0 tells the software to use its internal default, which is 1,980. Setting this item to any other value, overrides the software default and the corresponding AAL5 protocol data unit (PDU) is 8 octets larger than this setting.
`ifatm_cksum`	3 =compute TCP/UDP checksums on the board for both transmission and reception	Disables and enables TCP and UDP checksum operations on the board. The board can be configured to do checksum operations for reception only (1), transmission only (2), both (3), or none (0).
`ifatm_n_ifnets`	4 =four IP network interfaces (`atm0` to `atm3`) are created automatically during reboot	Number of logical IP network interfaces that are created automatically at reboot. Complete description of this configuration task is provided in Section 3.5.1, page 104.[5] Valid values are 1 through 128 inclusive.

To configure the IRIS IP-over-ATM driver, perform the following steps:

1. Edit the`/var/sysgen/master.d/if_atm` file. Change any of the settings summarized in Table 14, page 103. Close the file.
2. Use the `autoconfig` command, as illustrated below, to build a new driver into the operating system.

   ```
   % /sbin/su
   Password: thepassword
   # /etc/autoconfig
   ```

3. If this is the only or the last configuration task, reboot the system to start using the new operating system. Otherwise, perform the other configuration tasks, then reboot.

4 Compliant with RFC 1577 when using LLC/SNAP encapsulation.

5 To assign each of these IP network interfaces to a specific physical port, edit the appropriate file: for PVC use, edit /var/atm/pvc.conf file, as described in Section 3.5.5.1, page 109; for SVC use, edit the /var/atm/ifatm.conf file, as described in Section 3.5.5.2, page 111.

3.5 IP Network Interface Configuration

This section provides instructions for configuring the IP protocol stack through IP-over-ATM logical network interfaces.[6]

Note: This section assumes basic understanding and experience with IRIX IP configuration. Complete explanations for standard steps that are mentioned here are located in the online IRIS Insight document *The Advanced Site and Server Administrator's Guide* that is provided with each system.

The following steps must be performed before an IP application can send or receive over the IRIS ATM subsystem:

1. If more than one IP network interface is needed permanently, create the additional interfaces, as described in Section 3.5.1, page 104.
2. Edit the following network configuration files:
   - `/etc/hosts`, as described in Section 3.5.2, page 106.
   - `/etc/config/netif.options`, as described in Section 3.5.3, page 106.
   - optionally, `/var/sysgen/ifconfig-#.options`, as described in Section 3.5.4, page 107.
3. If IP is using SVCs, map each logical network interface to a physical port and provide the name of the subnetwork's ATM address resolution server by editing the `/var/atm/ifatm.conf` file, as described in Section 3.5.5.2, page 111.

   If IP is using PVCs, map remote IP addresses to virtual channel addresses, using the `/var/atm/pvc.conf` file, as described in Section 3.5.5, page 109.
4. Optionally, if IP is using SVCs, configure other LIS parameters, as described in Section 3.5.7.1, page 119.
5. To build the configuration changes into the system, reboot the system.

3.5.1 Increasing the Number of IP Network Interfaces

By default, the IRIS ATM driver creates four logical IP network interfaces (`atm0` to `atm3`), regardless of the number of IRIS ATM boards installed in the system.

6 Two communication paths to the IRIS ATM subsystem are available for upper-layer IP applications: (1) the standard BSD socket interface, and (2) a character device interface. Refer to the *IRIS ATM API Programmer's Guide*, publication SG–2235, for information about the character device interface.

For IP-over-ATM to work, there must be only one logical network interface for each IP subnetwork (LIS). Regardless of the number of installed IRIS ATM boards, the IRIS ATM driver can support up to 64 different logical network interfaces.

To increase the number of IP network interfaces, complete the following steps:

1. Edit the `/var/sysgen/master.d/if_atm` file, as described in Section 3.4, page 102.
2. Edit the `/etc/config/netif.options` file, as shown in the following example:
   - Change the following entry in the `/etc/config/netif.options` file:

     ```
     : if_num = 8
     ```

   - Change the entry to look like the following:

     ```
     if_num = #
     ```

     The # option specifies the total number of IP logical network interfaces for the system, including built-in Ethernets, and other options that support IP traffic.

   Note: The additional interfaces are created the next time the system is rebooted.

You can use the `netstat` command to verify that the additional network interfaces have been created and configured. In the following example, ten IRIS ATM network interfaces exist and the first three interfaces are configured and enabled; seven interfaces are disabled and not configured.

```
% /usr/etc/netstat -in
...
atm0 9180 netaddress IPaddress ...
atm1 9180 netaddress IPaddress ...
atm2 9180 netaddress IPaddress ...
atm3* 9180 none   none ...
atm4* 9180 none   none ...
atm5* 9180 none   none ...
atm6* 9180 none   none ...
atm7* 9180 none   none ...
atm8* 9180 none   none ...
atm9* 9180 none   none ...
...
```

3.5.2 Mapping Names to IP Addresses: The `/etc/hosts` File

In the local `/etc/hosts` (network address information) file, add one line for each logical ATM network interface. Each line that you add specifies the IP address and the name by which one logical network interface is known.

The following example contains entries for IP-over-ATM network interfaces on a machine called mars. The standard network portion (`netid`) of the IP addresses in this example is 190.15, while the locally assigned subnet portions (most significant byte of hostid) are 1 and 6, and the local host address portions are 10, 11, and 13.

```
190.15.1.10 atm0-mars.engr.cmpy.com atm0-mars#subnet 1
190.15.6.10 atm1-mars.engr.cmpy.com atm1-mars#subnet 6
190.15.1.11 atm2-mars.engr.cmpy.com atm2-mars
190.15.1.13 atm3-mars.engr.cmpy.com atm3-mars
190.15.6.11 atm4-mars.engr.cmpy.com atm4-mars
```

You may need to add the same entries to other databases located on other systems on the ATM networks. For example, you may need to add these entries to an ATM address resolution server's database, or an NIS database, or you may need to update the `/etc/hosts` file on every host on the ATM network.

3.5.3 Mapping IP Addresses to Network Interfaces: The `netif.options` File

In the `/etc/config/netif.options` file, add a pair of lines for each logical ATM network interface. The IRIS ATM driver supports up to 64 network interfaces with names like `atm0`, `atm1`, or `atm47`. To enable the maximum number of network interfaces, you add 64 line-pairs that have the same format as described in the following example.

To inititate the changes in the file, reboot the system.

> **Note:** Do not configure any IRIS ATM network interface as the primary one (that is, as `if1name`). ATM does not support broadcasting, which is required by many standard network and client/server services that operate over the primary network interface like BOOTP, NIS, RIP, OSPF, `timed`, `gated`, and the multicast version of NTP.

For the IRIS ATM network interface called `atm0`, the entry looks like the following:

```
if#name=atm0
if#addr=name
```

The *name* option is a network connection name (such as mars) or an IP address (such as 190.15.1.1) from the `/etc/hosts` file, and the pound sign (#) is replaced with any numeral, except 0, 1, or an already used numeral.

For the IRIS ATM network interface called `atm47`, the line looks like the following:

```
if#name=atm47
if#addr=name
```

The *name* option is a network connection name or IP address from the `/etc/hosts` file and the pound sign (#) is replaced with any numeral, except 0, 1, or an already used numeral.

3.5.4 Configuring Optional Operational Parameters: The `ifconfig-#.options` File

To configure the operational parameters for each network interface, create a file called `ifconfig-#.options` in the `/etc/config` directory. The # option of the file's name matches the interface's order in the `/etc/config/netif.options` file. For an interface enabled as `if2name=atm#` in the `netif.options` file, create a file called `ifconfig-2.options`; for an interface enabled as `if48name=atm#`, create a file called `ifconfig-48.options`. In most cases, this step is optional since the system has default settings for these parameters, as described in Table 15, page 108.

To initiate the changes in this file, use the `/usr/etc/ifconfig` command to disable and re-enable the interface.

> **Note:** These files are optional. If the file for an interface does not exist, the network interface is automatically configured with the defaults described in Table 15, page 108. If the file exists, but only specifies settings for some of the parameters, the system configures the unmentioned parameters with default settings.

For each IRIS ATM network interface, you can configure one or more of the parameters described in Table 15, page 108.[7]

[7] The `broadcast` and `arp` parameters do not apply to IRIS ATM network interfaces.

Table 15. Network Interface Parameters

Parameter	Default setting	Description
netmask (32-bits)	No mask; no subnets. The digits in the network portion of the Internet address are set to 1; the digits in the host portion of the Internet address are set to 0.	Value used to create two or more IP subnetworks from a single Internet network address.
route metric	0 The most favorable rating possible.	Hop count value advertised to other routers by the routing daemon (`routed`). Possible settings range from 0 (most favorable) to 16 (least favorable, infinite).
debug	Disabled.	When debugging is enabled, a wider variety of error messages are displayed when errors occur.

The following are examples of text that can be placed within `ifconfig-#.options` files:

- To specify a subnet mask for the interface:

  ```
  netmask 0x########
  ```

  The ######## option is the 32-bit mask in hexadecimal notation. The standard network portion and the locally designated subnet bits should be set to ones; the bits designated locally as host bits should be set to zeros. For example, 0xFFFFFF00 could be a subnet mask for a Class B address with 16 bits of standard network address, 8 bits of locally designated subnet, and 8 bits for host addresses.

- The following is an example line of a file that specifies a netmask and sets a route metric:

  ```
  metric 2 netmask 0xFFFFFF00
  ```

The `ifconfig`(1M) man page provides more details about the parameters.

Note: This configuration task can be done with the `ifconfig` command; however, the configuration is lost at the next reboot.

3.5.5 Mapping IP Interfaces to the ATM Subsystem

Before the IRIS ATM software can set up an ATM virtual channel for an IP network interface, the remote IP address must be resolved to an ATM address or a VPI/VCI value. The IRIS ATM subsystem handles this address resolution differently for PVCs and SVCs. Follow the set of instructions that are appropriate for your system's usage. If both PVCs and SVCs will be used, follow the instructions in both sections. For a description of how IP address resolution is done, see Section 1.6, page 32.

3.5.5.1 Address Resolution for PVCs

The IRIS ATM subsystem maintains in its memory an IP-to-VC address resolution table for use with PVCs. For each VC, the table maps an IP address to an ATM VPI/VCI value and a physical port. Every remote system with which this system will exchange data (either send to or receive from) must have an entry in this file. The `atmarp -f` utility is provided for loading this mapping file into the address resolution table.

Note: You do not need to add the PVCs used by ATM signaling and ILMI. The IRIS ATM software does this automatically.

Use the following instructions to create and load the IP-to-VC address resolution and port assignment table:

1. Create or open a `/var/atm/pvc.conf` file. In this file, you must add one line for every IP host with whom data is to be exchanged via PVCs. The maximum number of entries is 256. The maximum number of entries per port is 37.

2. For all remote IP hosts with whom data will be exchanged (sent or received), enter one line in the following format:

   IPaddress port# vpi vci flags

   The entries in each line have the following meaning:

   - *IPaddress.* An IP address in dotted decimal notation or its name as listed in the hosts file. If a host name is used, it must map to a numerical IP address in the `/etc/hosts` file. The address identifies a remote IP host with which this system wants to exchange data. The network portion of this IP address must match the subnetwork portion of one of the station's own IP-over-ATM logical network interfaces, as configured in the `/etc/config/netif.options` file. All traffic, incoming and outgoing, exchanged with this host IP address travels over the port

specified on this line. Remote hosts that have the same subnetwork address portion of their IP addresses can use different physical ports. For example, in the following sample `pvc.conf` file, hosts 255.86.8.3 and 255.86.8.11 both have netid 255.86.8; however, the VC for host 255.86.8.11 uses port 0 while that for 255.86.8.3 uses port 1.

- *port#.* Specifies the unit number for the IRIS ATM hardware over which the virtual channel to this host is established. For each *port#* there must be a functioning port.
- *vpi*. A virtual path address (up to 8 bits) in decimal or hexadecimal notation. For example, 0xA7 or 17. The user can select these values. However, the values must match at each endpoint on a link, as illustrated in Figure 12, page 19 and Figure 13, page 19.
- *vci*. A virtual channel address (up to 16 bits) in decimal or hexadecimal notation. For example, 0x104C or 4172. The user can select these values. However, the values must match at each endpoint on a link, as illustrated in Figure 12, page 19 and Figure 13, page 19.

  Note: The values VPI=0 with VCIs from 0 to 32 (decimal) are reserved for special uses, such as ILMI and signaling.
- *flags*. An option with which *n* is the only currently supported value for this field. *n*is used to inhibit use of 802.2 SNAP LLC headers on IP packets that are sent on this VC.

The following is an example of a `/var/atm/pvc.conf` file. The entries in this example indicate that the system has at least two IRIS ATM ports (port 0 , `/hw/atm/0` and port 1, `/hw/atm/1`) and four logical ATM network interfaces, which are configured with IP addresses 187.3.*x.x*, 187.4.*x.x*, 255.86.8.*x*, and 255.86.34.*x*.

```
#
# hostnameportVPIVCIflags
# unit#
# --------  ----  ---  ---  -----
atm-host3  0  0x1A0x32E
187.3.2.7  0  059
255.86.8.11  0  0x100x2A34  n
255.86.8.3     1  25565535
187.4.2.55  1  16148
255.86.34.11  1 18924830
```

3. Reboot or use the following command lines to load the mappings into memory.

   The following command line checks to see if `atmarp` is already running:

   ```
   # /sbin/ps -e | grep atmarp
   ```

   If `atmarp` is already running, use the following command line to interrupt it so that it restarts itself and loads the new file into memory:

   ```
   # /usr/bin/killall -Hup atmarp
   ```

   If `atmarp` is not running, use the following command line to start it so that it loads the new file into memory:

   ```
   # /usr/etc/atmarp -f  /var/atm/pvc.conf
   ```

   Note: Each invocation of `atmarp` spawns a process. Repeated invocations of `atmarp` create multiple `atmarp` processes that interfere with proper handling of the table. Before loading a new table, use the `ps` command to verify that no `atmarp` process is currently running. You must use the `kill` or `killall` command to remove or interrupt any current ones before starting a new one.

4. To subsequently make changes to the table, edit the file containing the IP-to-VC mappings and interrupt and restart the current `atmarp` process, as described in the previous step.

You can verify the entries of the currently loaded IP-to-VC address resolution table with the following command line:

```
# /usr/etc/atmarp -a
```

3.5.5.2 Address Resolution for SVCs

Follow these procedures to configure a system for SVC address resolution and port usage:

1. If your system is directly connected to an ATM system that does not do address registration, give your system its ATM address, as explained in Section 3.7.1, page 126.

   Note: Many ATM switches assign network prefixes (as required by the ATM UNI Signaling standard). Endpoints and some switches do not. The IRIS ATM ILMI daemon performs as an endpoint; it does not assign network prefixes to adjacent or local ATM interfaces.

2. Open a `/var/atm/ifatm.conf` file; if the file does not exist, create a `/var/atm/ifatm.conf` file.

3. For each of your system's logical network interfaces that is using IP over SVCs (that is, each LIS), indicate the port for that LIS by entering one line in the file.[8] The format for each entry is as follows:

   ```
   atm# port #
   ```

   The entries in each line have the following meanings:

   - `atm#`. The logical network interface name exactly as it appears in the `/etc/config/netif.options` file. There must be only one port entry for each interface (that is, one line for `atm0`, one line for `atm1`, and so on). Not all of the enabled logical network interfaces have to be listed; make entries only for those interfaces that carry SVCs. All of the IP-over-ATM network interfaces on a system must be members of the same LIS (that is, no two members of the same LIS can reside within the same Origin processor module).

     Note: This restriction is present only to enable automatic default and correct operation of the IRIX routing daemon (for example, `routed`). Nothing in the ATM protocol, the IRIS ATM software or hardware, or RFC 1577 requires this restriction. For RFC 1577 compliance, each IRIX ATM port can support one member of each LIS. For example, a module with 64 IRIX ATM ports could support 64 hosts from a single LIS. For ATM compliance, there are no restrictions regarding the number of logical network interfaces.

   - `port`. A keyword (required entry).
   - #. A port number (a decimal digit) for the ATM port as displayed by `hinv` or at `/hw/atm/#`. There can be multiple entries (lines) for a port. For example, a single port can service two or more logical network interfaces: `atm0` and `atm1` can both be using port 0.

4.

   For Origin 2000 or Onyx2 systems, signaling for each ATM port can be either Classic IP, which uses UNI signaling, or Fore IP, which uses Simple

[8] In IP-over-ATM environments, unlike Ethernet or FDDI, each physical port can serve numerous logical network interfaces (each with its own IP addresses). For more detail, see Section 1.6, page 32.

Protocol for ATM Network Signaling (SPANS). Only UNI is available for other systems.

Specify the `iftype` *type* parameter as either `iftype atm` (for UNI signaling) or `iftype spans` (for SPANS signaling). By default, *type* is `atm`. The following example specifies SPANS signaling:

```
atm0 port 0 iftype spans
```

For more information about this ATM 2.3.1 feature, see the `ifatmconfig`(1M) man page.

5. For each IP-over-SVC logical network interface on your system, obtain the ATM address of the subnetwork's ATM address resolution server. This information can usually be displayed on the server's terminal.

   Note: If you want to configure this station as the ATM address resolution server for a subnetwork, skip this step and the next one for that particular network interface. Proceed with the other steps in this section, complete the rest of the installation and configuration, and bring the system into operation so that the IRIS ATM software obtains its ATM NSAP address. Then, return to these instructions and complete this step and the next one for the skipped network interface(s). You can display the local system's ATM address for each network interface with the following command line:

   ```
   # ifatmconfig atm#
   ```

6. To each of the lines created above, append the ATMARP server for that subnetwork (LIS). Now, the complete format for each line is as follows:

   `atm# port # arpserver` *ATM_address*

   The entries in each line have the following meanings:

   - `atm#`, `port`, and #. Explained in the previous step.
   - `arpserver`. A keyword (required entry).
   - *ATM_address*. A 20-byte ATM NSAP or an up to 15-byte native E.164 address in hexadecimal format. See Table 16, page 114, for examples of acceptable formats; Figure 11, page 17, and Figure 10, page 16, show the address. For a description of this address, see Chapter 1, page 1, or the glossary entry for *ATM address*.

Note: To make this station serve as the ATMARP server for a subnetwork, specify the local port's own ATM address on the line for that interface. If the port does not yet have an ATM address, skip this step and return to it later.

- Text to the right of a pound sign (#) is ignored, that is, the text is treated as a comment.

7. If this is your only or last configuration task, use the command lines below to start using the new configuration. Otherwise, perform the other configuration tasks and reboot the system.

```
# /etc/ifconfig atm#   down
<do this for each atm interface listed in the file>
# /usr/etc/ifatmconfig -F /var/atm/ifatm.conf
# /etc/ifconfig atm# up
<do this for each atm interface listed in the file>
```

or

```
# reboot
```

Table 16. Formats for ATM NSAP Address in the `ifatm.conf` File

Example	Comments
0x39.0840.00112233445566778899.0d0001220033.00	Dots between fields of address.[9]
0x39.08.40.00.11.22.33.44.55.66.77.88.99.0d.00.01.22.00.33.00	Dots between all bytes.
0x390840001122334455667788990d000122003300	No dots.

Figure 29, page 115, shows an example of this file for a system that has four logical IP-over-ATM network interfaces using SVCs (`atm0`, `atm1`, `atm7`, `atm8`) and at least three IRIS ATM ports (port 0, port 1, port 3). (This system probably has a unit 2 ATM board that is not listed in this file, meaning that port 2 does not communicate with networks through IP-over-SVCs.) The IP addresses listed

9 The address must contain 40 hexadecimal characters (20 bytes); the periods (dots) do not count as characters.

All fields, when separated by dots, must contain an even number of characters.

All bytes must contain two hexadecimal characters. It is not legal to strip leading zeros. Use 0x0F.08; do not use 0xF.8.

after the pound signs are included to facilitate identification; they are not necessary.

```
# ATM address resolution server for each IP‹over‹ATM network interface using SVCs
atm0 port 0 arpserver 0x39.0840.080ffe1000000f11509d.00d904805989.00      #for 223.10.20.2
atm1 port 1 arpserver 0x47.0005.80.ffe100.0000.f115.098d.00d90480598A.00  #for 223.10.71.15
atm7 port 3 arpserver 0x45.000014083262189F.0000.098d.00d9048059f2.00     #for 223.10.98.33
atm8 port 3 arpserver 0x45.000014083262189F.0000.098d.00d907A16CCC.00     #for 223.10.52.1
```

a11516

Figure 29. Format for `ifatm.conf` File

3.5.6 Configuring a System to Run SPANS (Origin2000 and Onyx2 Systems Only)

The following sections provide the following SPANS information:

- How to configure a system to run the Simple Protocol for ATM Network Signaling (SPANS)
- How to verify SPANS operation
- SPANS implementation limitations

3.5.6.1 Configuring for SPANS

Use the following procedure to configure SPANS:

Procedure 1: SPANS Configuration Procedure

1. After installing ATM 2.3.1, update all of the following configuration files with their new versions:

   ```
   /var/atm/ifatm.conf
   /var/atm/spansd.conf
   /var/sysgen/master.d/if_atm
   /var/sysgen/master.d/if_atmarp
   ```

2. Change the following configuration files, as noted:

   - `/var/atm/atmilmid.conf`

     To disable ILMI on the ports, comment out all lines beginning with `ATMPORT` and `ATMADDRESS`.

   - `/var/atm/atmsigd.tcl`

To disable the UNI signaling daemon on the ports, comment out all lines beginning with `buildstack`.

- `/var/atm/ifatm.conf`

  To disable Classic IP on all ports, comment out all lines beginning with `atm`*x* (where X is 0, 1, 2, 3, etc). Next, to place each port into SPANS mode, add a line as follows for each port:

  ```
  atm0 port 0 iftype spans
  ```

- `/var/atm/spansd.conf`

  Depending on whether you will be using the SPANS load balancing feature or you want to change the default settings for any of the SPANS configuration parameters, you might have to edit this file. To determine this, see the `spansd`(1M) man page.

- `/etc/config/netif.options`

  Edit this file to include any of the ATM interface that you want to automatically configure up at system startup time.

3. Configure your FORE ATM switch for SPANS. Each port that will run SPANS must have UNI signaling disabled and SPANS signaling enabled.
4. Log in to the ATM switch's "ATM Management Interface" (AMI).
5. To disable UNI signaling on these ports, go into the `localhost::configuration uni` level and use the `delete` command to delete each port that you want to run SPANS, as follows:

   ```
   delete 1a1 0
   ```

6. To enable SPANS signaling on these ports, go into the `localhost::configuration spans` level and on each port that you want to run SPANS, use the `new` command, which also changes the default AAL from 4 to 5 and the maximum number of VCI's on the port to 256 (0-255). This step is required in this implementation of SPANS (see `spansd`(1M). Following is an example ot the `new` command:

   ```
   new 1a1 0 -aal 5 -maxvci 255
   ```

7. Configure the kernel by using the `autoconfig`utility, and then reboot your system.

3.5.6.2 Verifying SPANS Operation

To verify that the SPANS protocol has initialized properly, view the `/tmp/spansd.log` file . Included in the messages are the settings for the SPANS timers, the load balance grouping, the size of the ATMARP table, and the switch addresses assigned to each of the ports. If the interface initialized properly, the file should end with messages like those in the following examples:

```
INFO: atm0 connected to switch port f20f1abf.08
INFO: atm1 connected to switch port f20f1abf.09
INFO: atm2 connected to switch port f20f1abf.10
INFO: atm3 connected to switch port f20f1abf.11
```

If these messages do not appear, perform the following troubleshooting steps:

Procedure 2: SPANS Troubleshooting

1. Enter the following command:

   `ifatmconfig atmX`
   `X` is the port number of each of the ports that should be running SPANS

2. Verify that the output of this command shows that the port is configured with an *iftype* of `SPANS`. If not, check the `/var/atm/ifatm.conf` file to ensure that it is correct. Also, check the other `/var/atm` files to insure that they are set up according to Procedure 1, page 115.

3. Verify that the `/tmp/spansd.log` file exists. If not, the SPANS daemon, `spansd`, might not have started or it might have died.

4. Use the `ps -el | grep spansd` command to check the process table to verify that the SPANS daemon, `spansd`, is running.

5. Enter the `netstat -I atmX 1` or `netstat -C` command, and view the SPANS status messages and their replies to verify that each ATM port that is running SPANS is receiving and transmitting approximately 2 packets per second. If this is not happening, verify the ATM switch configuration.

6. If all of the above appears to be working properly, attempt to contact a remote host that is directly attatched to the ATM network by using the `ping` command. This other host MUST also be running the SPANS protocol. If this is not successful, use the `atmarp -a` command to determine if, at least, the ARP and ATM signaling were successful. The output of `atmarp -a` should display for the remote host a line that shows a valid interface, AAL, VP, and VCI. The `Flags`field should show `SPANS,Complete`. If this is not the case, contact techincal support for assistance.

3.5.6.3 SPANS Implementation Limitations

The implementation of the SPANS protocol has the following limitations:

- The SPANS daemon does not support Multicast or Multi-Point-to-Point (MPP) channels.
- The FORE ATM switch must be configured to run SPANS over AAL 5. This is not the default setting in the ATM switch.
- The FORE ATM switch must be configured to use a maximum of 256 VCI's. This is not the default setting in the ATM switch.
- The SPANS daemon does not support host-to-host connections. There must always be an ATM switch connecting the hosts together.
- The SPANS daemon is supported only on Quad OC-3c XIO ATM hardware.
- The `pending entry` and `complete entry` timers are implemented as two separate asynchronous timers. The `pending entry` timer is running in the kernel and has an interval of ten seconds. The `complete entry` timer is running in the SPANS daemon and has an interval of one minute. Because of their asynchronous nature, these timers can be off by as much as +/- one interval. For the `pending entry` timeout, this means that a timeout can occur +/- ten seconds from the specified timeout value. For the `complete entry` timeout, this means that a timeout on an "idle complete entry" can occur +/- one minute and ten seconds from the specified timeout value.

3.5.7 Configuring LIS Parameters

Each IP-over-ATM logical network interface (endpoint for one LIS), has the following configurable parameters:

- The physical port associated with the LIS (instructions provided in Section 3.5.5.2, page 111).
- The ATMARP server for the LIS (instructions provided in Section 3.5.5.2, page 111).
- The transmission rate for SVCs to that LIS (instructions provided in Section 3.5.7.1).
- The VC timeout used for determining when an SVC is torn down due to inactivity (instructions provided in Section 3.5.7.1).

Note: For temporary (or dynamic) configuration of LIS parameters, see the `ifatmconfig`(1M) man page.

3.5.7.1 Set Transmission Rate and Timeout

Follow the instructions in this section to configure non-default LIS operational parameters, such as the transmission rate used on the VCs for that LIS, and the timeout used for inactive VCs. These configurations are optional; the IRIS ATM software contains default settings that are used if you do not configure these items.

1. Open the `/var/atm/ifatm.conf` file.
2. For each LIS for which you want to specify a non-default peak transmission rate, add a line, as shown in the following format. If this line does not exist for an LIS, the IRIS ATM software uses a default rate of 135,991,460 bits per second.

   `atm# vcrate` *bits_per_second*

   The entries in each line have the following meanings:

   - `atm#`. The logical network interface name exactly as it appears in the `/etc/config/netif.options` file. There must be only one rate for each logical network interface (that is, one vcrate for `atm0`, one vcrate for `atm1`).
   - `vcrate`. A keyword (required entry).
   - *bits_per_second*. The upper–layer payload for the desired transmission rate. (For example, for the mezzanine HIO board, this can be any of the transmission rates from Appendix B, page 211.) The driver uses this value to calculate the transmission rate for VCs that are opened for the associated LIS. The resulting traffic parameter will be `BEST-EFFORT` with a peak cell rate (CLP=0+1) of *bits_per_second*/384.

     Note: The values in Appendix B, page 211, are expressed in megabits-per-second. You must convert the values to bits-per-second before entering them into this field. If the value entered in this field does not exactly match a configurated rate queue on the board, the software will use the closest match.

3. For each LIS for which you want to specify a non-default timeout, add a line in the following format. If this line does not exist for an LIS, the IRIS ATM software uses a default timeout of 20 minutes.

`atm# vctimeout` *minutes*

The entries in each line have the following meanings:

- `atm#`. The logical network interface name exactly as it appears in the `/etc/config/netif.options` file. There must be only one timeout for each logical network interface (that is, one VC timeout for `atm0`, one VC timeout for `atm1`).
- `vctimeout`. A keyword (required entry).
- *minutes*. The number of minutes that can pass during which no data is transmitted or received on an SVC before the VC is torn down.

  Note: The *vctimeout* does not affect the permanently open SVC that is created for communicating with the ATMARP server.

4. If this is your only or last configuration task, use the following command lines to start using the new configuration. Otherwise, perform the other configuration tasks, then reboot the system.

```
# /etc/ifconfig atm# down
<do this for each atm interface listed in the file>
# /usr/etc/ifatmconfig -F /var/atm/ifatm.conf
# /etc/ifconfig atm# up
<do this for each atm interface listed in the file>
```

or

```
# reboot
```

3.5.7.2 The `ifatmconfig` Utility

The `/usr/etc/ifatmconfig` utility is provided for on-the-fly configuring of IP-over-SVC parameters, such as the address of the ARP server for each LIS, the timeout period for tearing down inactive VCs, and the transmission rate for SVCs to the LIS.

- Dynamic Configuration of ATMARP Server

  To change an ATM address resolution server after system startup, disable each network interface for which you are going to make the change, and invoke the `ifatmconfig` utility for each new server. The command line requires the following format:

```
# /usr/etc/ifconfig atm# down
# /usr/etc/ifatmconfig atm# port # arpserver NSAP_address
# /usr/etc/ifconfig atm# up
```

Note: See Table 16, page 114, for valid formats of the *NSAP_address*.

- Dynamic Configuration of SVC Transmission Rate

  To change the peak transmission rate used for SVCs created for an LIS, use the command lines below:

  ```
  # /usr/etc/ifconfig atm# down
  # /usr/etc/ifatmconfig atm# vcrate bits_per_second
  # /usr/etc/ifconfig atm# up
  ```

  The *bits_per_second* option indicates the highest rate the software will use in its traffic contract. The resulting SVCs will have peak cellrates set to *bits_per_second*/384.

  Note: The IRIS ATM signaling and ILMI software creates two best-effort PVCs per port for use in communicating with the adjacent switch. This overhead traffic is sporadic and uses only a portion of any rate queue's bandwidth; however, the higher the configured rate, the larger the percentage of the port's total bandwidth that can be occupied by overhead whenever there is overhead traffic to transmit.

- Dynamic Configuration of Timeout for Inactive VCs

  To change the timeout used for tearing down inactive VCs, use the following command line:

```
# /usr/etc/ifconfig atm# down
# /usr/etc/ifatmconfig atm# vctimeout minutes
# /usr/etc/ifconfig atm# up
```

3.6 IRIS ATM Signaling Protocol Stack Configuration

This section provides instructions for configuring the IRIS ATM signaling software. In this section, the required procedure is in Section 3.6.1, page 123.

The IRIS ATM signaling software (`atmsigd`) is the collection of modules that manages the protocol stack for each ATM user-network interface (UNI) and the interface to the IRIS ATM driver, as explained in Section 1.4.3, page 20, and shown in Figure 18, page 29. Before starting IRIS ATM, you must configure `atmsigd` to build one UNI stack for each IRIS ATM physical connection (port). You do this by editing the `/var/atm/atmsigd.tcl` file. The following is an example of the portion of the file that you must edit. These are the lines as they are shipped with the product:

```
buildstack 1 /hw/atm/0 AF30 AF30
# buildstack 1 /hw/atm/0 AF30 AF31
# buildstack 1 /hw/atm/0 AF31 AF31

# buildstack 2 /hw/atm/1 AF30 AF30
# buildstack 2 /hw/atm/1 AF30 AF31
# buildstack 2 /hw/atm/1 AF31 AF31

# buildstack 3 /hw/atm/2 AF30 AF30
# buildstack 3 /hw/atm/2 AF30 AF31
# buildstack 3 /hw/atm/2 AF31 AF31
```

The format for each line is as follows:

`# buildstack` *identification*`# /hw/atm/#` *version_SSCOP version_Q.2931*

The items have the following meanings:

- The leading #. Indicates that the line is commented out (unreadable). You must remove this character to make the line active (readable).
- `buildstack`. A keyword (required entry).
- *identification*#. Identifies the UNI protocol stack for that port. Valid values are all non-zero decimal numerals. For each numeral, only one line should be readable. It is recommended that these numerals be used sequentially.

- /hw/atm/#. Specifies the ATM port (for example, `/hw/atm/0` identifies ATM port 0).
- *version_SSCOP*. Specifies the ATM UNI version for the service specific convergence protocol layer. The valid entries are either `AF30` for ATM UNI Specification 3.0 [Q.SAAL1 and 2] or `AF31` for ATM UNI Specification 3.1 [Q.2110]. This version must match the version used on the adjacent switch.
- *version_Q.2931*. Indicates the ATM UNI version for the signaling protocol (that is, the Q.2931 layer). The valid entries are either `AF30` for ATM UNI Specification 3.0 or `AF31` for ATM UNI Specification 3.1.

3.6.1 Required `atmsigd` Configuration

Follow these steps to perform the standard, required configuration of the IRIS ATM Signaling daemon:

1. Open the `/var/atm/atmsigd.tcl` file.
2. Uncomment, that is, remove, only one line for each ATM port (`/hw/atm/#`).
3. If your system has more than 3 ATM ports, create one new line for each additional port. Each line must have a unique identification number (*identification*#) and device name (`/hw/atm/#`).

   For example, to configure a system with 4 ATM ports where two ports are using ATM UNI version 3.0 for both configurable layers, one is using 3.0 for SSCOP and 3.1 for the signaling, and one is using 3.1 for SSCOP and signaling, edit the file to look like the following example:

```
buildstack 1 /hw/atm/0 AF30 AF30
# buildstack 1 /hw/atm/0 AF30 AF31
# buildstack 1 /hw/atm/0 AF31 AF31

buildstack 2 /hw/atm/1 AF30 AF30
# buildstack 2 /hw/atm/1 AF30 AF31
# buildstack 2 /hw/atm/1 AF31 AF31

# buildstack 3 /hw/atm/2 AF30 AF30
buildstack 3 /hw/atm/2 AF30 AF31
# buildstack 3 /hw/atm/2 AF31 AF31
buildstack 4 /hw/atm/3 AF31 AF31
```

4. If this is your only configuration task, use the following command lines to restart `atmsigd`. Otherwise, continue with your tasks and these changes

will take effect the next time the IRIS ATM software is started. For example, when the system is rebooted or when `/etc/init.d/atm start` is invoked.

```
# /etc/init.d/atm stop
# /etc/init.d/atm start
```

3.6.2 Optional `atmsigd` Configuration

If an adjacent switch does not use the standard VPI/VCI address (VPI=0, VCI=5) for its ATM signaling communications, use the following instructions to change the value used on that port:

1. Open the`/var/atm/atmsigd.tcl` file.
2. Locate the line for the port in question.
3. Add a pound sign to comment out the standard `buildstack` line, as shown below:

   ```
   # buildstack 3 /hw/atm/2 AF30 AF31
   ```

4. Add lines in the following format:

   `build aal5atm` *identification*`# /hw/atm/#` *VPI VCI*
   `build qsaal` *version_SSCOP identification*`#`
   `build q93b` *version_Q.2931 identification*`#`

   For example, to set VPI=2, VCI=8 on the port used in the preceding example for commenting out the `buildstack` line, the additional lines look like the followng:

   ```
   build aal5atm 3 /hw/atm/2 2 8
   build qsaal AF30 3
   build q93b AF31 3
   ```

Note: Alternatively, you may create your own procedure in the file, modeled after the `buildstack` procedure, then invoke that procedure with the arguments. This file must be written in the Tool Command Language (TCL) scripting language.

3.6.3 Disabling `atmsigd`

For configurations that do not require `atmsigd` (for example, PVC-only environments), rename the daemon's configuration file so that the process

terminates itself almost immediately after startup. The following command lines are an example of this procedure:

```
# mv /var/atm/atmsigd.tcl /var/atm/atmsigd.tcl.O
```

3.6.4 Running `atmsigd` in Debug Mode

If needed, the IRIS ATM signaling daemon (`atmsigd`) can be run in an interactive mode for debugging. When started in this mode, `atmsigd` responds to commands (phrased in TCL syntax) from the terminal (stdin). In this mode, you can manipulate the following items on the fly:

- Amount of error reporting for each layer of the signaling stack. Three levels are available: `ERROR` (lowest level), `TERSE`, and `VERBOSE`.
- VPI/VCI address for the permanent virtual circuit over which signaling occurs. This address is configured into the AAL5 layer.
- The ATM UNI version for the SSCOP and Q.2931 layers of the signaling stack.
- Creation (setup) and deletion of (teardown) virtual channels.

Caution: Running `atmsigd` in interactive mode can easily result in dysfunctional ATM stack configurations. Use this mode with caution.

Complete instructions are available in the `atmsigd`(1M) online man page.

3.7 IRIS ATM Interim Local Management Interface Configuration

This section provides instructions for configuring the IRIS ATM interim local management interface (ILMI) software. In this section, the only required procedure is Section 3.7.1, page 126. Each time an IRIS ATM board is installed or removed, these required procedures must be performed.

The ILMI daemon (`atmilmid`) is the module that manages ATM address registration for switched virtual circuits (SVCs), and management, configuration, and control information for switched and permanent virtual circuits, as explained in Section 1.4.3.3, page 29. You must configure the ILMI daemon before it functions.

The ILMI configuration files are `/var/atm/atmilmid.conf` and `/var/atm/atmilmid.options`. The `atmilmid.conf` file configures one instance of the ATM user-network interface (UNI) for each physical port. The

`atmilmid.options` file sets runtime variables for the ILMI daemon. The new settings take effect when `atmilmid` is restarted manually or restarted automatically at the next reboot.

3.7.1 Required `atmilmid` Configuration

Perform the following steps to configure the ILMI daemon, `atmilmid`:

1. Open the `/var/atm/atmilmid.conf` file.
2. Each `ATMPORT` line in this file defines a VPI/VCI address for `atmilmid` to use in communicating over an ATM port (physical connection). Each entry in this configuration file has the following format:

   `ATMPORT` *port_index port_name VPI VCI*

   The items have the following meanings:

   - `ATMPORT`. A required word.
   - *index*. A non-zero, positive, unique-within-this-file integer that uniquely identifies each port. This number is used in the ATM MIB's object identification address (OID) to differentiate between ATM UNIs (ports). The number is used, when querying the MIB, to indicate which port on the system is being referenced. The number is independent of all other identification numbers used by IRIS ATM software (for example, the stack *identification*# used by `atmsigd`). The simplest procedure is to use 1 for `/hw/atm/0`, 2 for `/hw/atm/1`, and so forth.
   - *port_name*. A path that identifies an existing ATM port on the system. Each port can be mentioned only once in this file. For example, use `/hw/atm/0` for port 0, `/hw/atm/1` or port 1, and so forth.
   - *VPI* and *VCI*. Decimal numbers that indicate the well-known virtual path and virtual channel identifiers for the PVC over which the communication between the `atmilmid` and the adjacent ILMI module (for example, on the switch) takes place. It is highly recommended that you use the following values specified by the ATM UNI standards: VPI=0, VCI=16.

3. Each `ATMADDRESS` line in this file defines an ATM address (either ATM NSAP or native-E.164) for one ATM port (physical connection). This line is required only for ports that are connected to an ATM system that does not do ILMI address registration (for example, a switch that does not assign ATM addresses).

Note: Most ATM switches assign ATM addresses to their adjacent endpoints (as required by the ATM UNI Signaling standard). The addresses are either the network prefix portion of an ATM NSAP or a native-E.164 address. Endpoints (the user side of the ATM UNI) and some switches do not perform this task. The IRIS ATM software performs only as an endpoint, so its ILMI daemon does not assign ATM addresses to adjacent systems; it registers the local portions (ESI and SEL) for an ATM NSAP address and accepts an address assignment for an ATM NSAP or a native E.164 address.

Each entry in this configuration file has the following format:

`ATMADDRESS` *port_index address*

The items have the following meanings:

- `ATMADDRESS`. A required word.
- *port_index*. One of the *port_index* numbers used in an `ATMPORT` definition line. The number denotes which of the ATM ports will be configured with this address.
- *address*. The port's ATM address in hexadecimal notation. This address can be either an ATM NSAP or a native (non-NSAP) E.164 address. Do not use a prefix to the string of hexadecimal characters (for example, use FF, not 0xFF) and do not use separators (for example, use FFFF, not FF:FF). The following two examples show valid formats:

  A 20-byte ATM NSAP address:

  47000580ffe1000000f115098d080069042a4f00

  A native-E.164 address:

  4085551212

  Note: See the glossary items *ATM address*, *ATM NSAP address*, and *native E.164 address* for further explanation.

4. If this is your only configuration task, restart `atmilmid`. Otherwise, continue with your tasks and these changes will take effect the next time the IRIS ATM software is started. For example, when the system rebooted or when `/etc/init.d/network start` is invoked.

The following sample ILMI configuration file entries configure an ILMI daemon to listen for and respond to SNMP commands on four ATM physical connections (`/hw/atm/0` to `/hw/atm/3`). Each physical connection has a

permanent virtual circuit that uses 0 for its VPI and 16 (decimal) for its VCI. The file also provides an ATM NSAP address for port 1 (`/hw/atm/1`) and assumes that the addresses for the other ports will be supplied by the adjacent switches.

```
ATMPORT  1  /hw/atm/0  0  16
ATMPORT  2  /hw/atm/1  0  16
ATMPORT  3  /hw/atm/2  0  16
ATMPORT  4  /hw/atm/3  0  16
ATMADDRESS 1 47000580ffe1000000f115098d080069042a4f00
```

3.7.2 Optional `atmilmid` Configuration

A number of operational parameters for the ILMI daemon are configured into the daemon at runtime (for example, during a reboot). To change the default settings, create the `/etc/config/atmilmid.options` file. The parameters that you can configure in this file are listed in Table 17, page 128. The table also indicates the default setting for each parameter.

Table 17. Operational Parameters for ILMI Daemon

Parameter	Description	Default setting
socket (*-p*)	The address of the UDP socket on which the `atmilmid` listens as a subagent for requests from the main SNMP process (for example, from a MIB viewing application). If you reset this value, be sure to select an unused one.	23,849
loglevel (*-l*)	Level of error message logging. Valid values are: `DEBUG` (most errors reported), `INFO`, `NOTICE`, `WARNING`, `ERR`, `CRIT`, `ALERT`, and `EMERG` (least errors reported).	ERR

To change any of these parameters, use the appropriate instructions as follows:

- To change the UDP socket at which `atmilmid` listens for queries from the main SNMP agent, place this entry in the `atmilmid.options` file:

  `-p` *socket_number*

  The *socket_number* option is an unused UDP socket. You must also edit the `/etc/snmpd.remote.conf` file to include this socket number.

- To change the level of error message reporting, place this entry in the file:

  `-l` *loglevel*

  The *loglevel* option is one of the following words listed from the highest amount of reporting to the least amount of reporting: `DEBUG`, `INFO`, `NOTICE`, `WARNING`, `ERR`, `CRIT`, `ALERT`, `EMERG`. Each level reports all messages at its level and messages of all higher levels. For example, `ALERT` reports alert and emergency messages; `EMERG` only reports emergency ones; and `DEBUG` reports all possible messages. These messages are written into the `/var/adm/SYSLOG` file.

3.7.3 Running `atmilmid` in Debug Mode

The ILMI daemon can be manually invoked to operate in debug mode. By invoking the command with various options, you can specify the location for the error messages and what type of information to provide.

Complete information is available in the `atmilmid`(1M) online man page.

3.7.4 Verifying Location of ATM MIB Definition File

Before the contents of an ATM MIB can be viewed with an SNMP viewer, the ATM MIB definition file (`atmf_ilmi.mib`) must exist in the directory where the viewer application expects to find it. For example, the IRIXPro (or NetVisualyzer) Browser application expects the file to be in the `/usr/lib/netvis/mibs` directory. When IRIS ATM software is installed, it automatically places the ATM MIB definition file in the `/usr/lib/netvis/mibs` directory.

3.8 Summary of IRIS ATM Files

Table 18, page 130, lists the files specific to the IRIS ATM product, and describes the purpose for each file. This listing does not include standard IRIX files (such as `/etc/hosts` and `/etc/netif.options`) that affect the configuration and performance of IRIS ATM. The table indicates which files must be edited before IRIS ATM is functional (R), which files can optionally be edited to alter default settings (O), and which files should not be edited.

Table 18. Summary of IRIS ATM Files

File full path	Purpose	Edit
`atm` `/var/sysgen/master.d/atm`	Configure hardware driver for IRIS ATM-OC3c HIO board. For example, configure method for unit number assignment.	O[10]
`atm` `/etc/init.d/atm`	Script that initializes and starts the IRIS ATM subsystem including the hardware, ILMI and signaling software, and LIS for IP-over-ATM. Invoked during startup by a symbolic link in the `/etc/rc2.d` directory.	N
`atm.sm` `/var/sysgen/system/atm.sm`	Instruct `autoconfig` when it is building ATM driver for HIO board into the operating system.	N
`atmf_ilmi.mib` `/usr/lib/netvis/mibs/atmf_ilmi.mib`	For SVCs only: define the ATM ILMI MIB.	N
`atmhw.conf` `/var/atm/atmhw.conf`	Instruct `atmconfig -F` when configuring ATM hardware. For example, source for transmit clock.	O
`atmhw.conf-#` `/var/atm/atmhw.conf-#`	Instruct `atmconfig -F` when configuring hardware for one specific ATM port.	O
`atmilmid.conf` `/var/atm/atmilmid.conf`	For SVCs only: configure ATM ILMI software (`atmilmid`).	R
`atmilmid.options` `/etc/config/atmilmid.options`	For SVCs only: set optional runtime parameters for `atmilmid.`	O
`atmsigd.tcl` `/var/atm/atmsigd.tcl`	For SVCs only: configure ATM UNI stack for the signaling software (`atmsigd`).	R
`if_atm` `/var/sysgen/master.d/if_atm`	Configure IRIS ATM TCP/IP driver.	O
`ifatm.conf` `/var/atm/ifatm.conf`	For SVCs only: instruct `ifatmconfig -F` when configuring IP-over-ATM logical network interfaces. Configures address resolution server for each LIS and designates a port for each logical network interface to use.	R

File full path	Purpose	Edit
`network.atm` `/etc/init.d/network.atm`	For PVCs only: script that initializes IP-over-PVC connections. Invoked during startup by a symbolic link in the `/etc/rc2.d` directory.	N
`pvc.conf` `/var/atm/pvc.conf`	For PVCs only: instructs `atmarp -f` to create PVCs. Maps IP addresses to VPI/VCI addresses, designates a port for each IP address to use, and defines use of LLC/SNAP encapsulation.	R
`quadoc3` `/var/sysgen/master.d/quadoc3`	Configure hardware driver for IRIS ATM-OC3c 4Port XIO board.	N
`quadoc3.sm` `/var/sysgen/system/quadoc3.sm`	Instruct `autoconfig` when it is building ATM driver for XIO board into the operating system.	N
`sigtest.c` `/usr/lib/atm/examples/sigtest.c`	Example of IRIS ATM application programming interface implementation using the SVC commands.	N

3.9 Stopping and Restarting IRIS ATM

IRIS ATM consists of multiple modules in addition to the hardware. During operation, these parts need to stay synchronized with each other. Because of this, it is recommended that you use discretion and follow the steps recommended in Table 19, page 132, to reset, stop, or start IRIS ATM software or hardware. In general, follow these guidelines:

- It is always safe to use the `ifconfig` command to bring logical network interfaces up or down.
- Use the `/etc/init.d/atm` script to stop and restart the IRIS ATM software.

Table 19, page 132, provides suggested steps for smoothly stopping and starting IRIS ATM for some common tasks.

10 O = editing is optional; N = do not edit this file; R = editing is required for the IRIS ATM subsystem to become functional

Table 19. Stopping and Starting IRIS ATM Modules and Hardware

Task	

To reconfigure or restart one IRIS IP-over-ATM logical network interface.	Edit configuration files, if changes are desired. Enter the following commands:
	`ifconfig atm #down`
	`ifconfig atm # up`

To reconfigure parameters for LIS or to restart one LIS.	Edit a configuration file that has entries for this LIS only. Enter the following commands:
	`ifconfig atm # down`
	`ifatmconfig -F<`*filename*`>`
	`ifconfig atm # up`

To reconfigure the IP-over-ATM driver or restart the IRIS ATM driver.	Edit configuration file, if changes are desired. Enter the following commands:
	`ifconfig atm <`*each_one*`> down`
	`/etc/init.d/atm stop`
	`/etc/init.d/atm start`
	`ifconfig atm <`*each_one*`> up`

To reconfigure or restart the IRIS ATM ILMI software.	Edit configuration files, if changes are desired. Enter the following commands:
	`/etc/init.d/atm stop`
	`/etc/init.d/atm start`

To reconfigure or restart the IRIS ATM Signaling software.	Edit configuration files, if changes are desired. Enter the following commands:
	`/etc/init.d/atm stop`
	`/etc/init.d/atm start`

To reload the IP-over-PVC address resolution table.	Edit configuration file, if changes are desired. Enter the following commands: `killall -Hup atmarp`

Task	

To reconfigure or restart one IRIS ATM port without disrupting other ATM ports.[11]	Edit configuration file, if changes are desired. Enter the following commands:
	`ifconfig atm # down`
	Enter the preceding command for every interface on the port.
	`atmconfig -i` *port#* `-r`
	`atmconfig -i` *port#* `-d`
	`atmconfig -i` *port#* `-F /var/atm/atmhw.conf- #`
	`atmconfig -i` *port#* `-u`
	`ifconfig atm`<*each_interface_on_port*> `up`
	`killall -Hup atmilmid`

To reset and bring up (into operation) one already installed IRIS ATM port without disrupting other ATM ports.	`ifconfig atm # down`
	Enter preceding command for every interface on the port.
	`atmconfig -i` *port#* `-r`
	`atmconfig -i` *port#* `-d`
	`atmconfig -i` *port#* `-F/var/atm/atmhw.conf- #`
	`atmconfig -i` *port#* `-u`
	`ifconfig atm` <*each_interface_on_port*> `up`
	`killall -Hup atmilmid`

To reconfigure or restart a port that has PVCs.	`/etc/init.d/network.atm stop`
	`/etc/init.d/atm stop`
	`/etc/init.d/atm start`
	`/etc/init.d/network.atm start`

11 With the IRIS ATM-OC3c 4Port XIO board, two ports are reset with this command: either the upper-two ports or the lower-two ports (as organized on the board's I/O panel plate). Whenever either of a pair is specified for reset on the command line, both ports are reset.

Monitoring the IRIS ATM Subsystem [4]

This chapter describes procedures for monitoring the operation of an IRIS ATM subsystem.

4.1 Checking the Status of IRIS ATM

A number of commands are provided to help monitor the IRIS ATM subsystem, as shown in the following list:

- The `atmconfig` command (`-s` and `-m` options) provides information about the hardware (`-i#` specifies the port).
- The `atmstat` command displays status and performance statistics (`-i#` specifies the port).
- The `ifatmconfig` command displays logical IP subnetwork (LIS) information, such as an ATM address for the local endpoint, the ATMARP server, VC timeout, and transmission rate (`atm#` specifies the LIS, that is, the logical network interface).

Table 20, page 135, summarizes the information that can be displayed and the command line for each.

Table 20. Summary of IRIS ATM Status Information Displays

	Information	Command	More Info
Hardware Information	Local MAC address	`atmconfig -i# -m`	Section 4.1.1.9, page 146
	Local port's ATM address	`ifatmconfig atm#`	Section 4.1.1.10, page 146
	Current rates for the transmission rate queues on IRIS ATM HIO board (only)	`atmstat -i# -q`	Section 4.1.1.4, page 140

	Information	Command	More Info
	Board's configuration	`atmconfig -i# -s`	Section 4.1.1.1, page 137
	Version of the firmware currently loaded onto board	`atmconfig -i# -V`	Section 4.1.1.2, page 139
	State of IRIS ATM board (up, down, etc.)	`atmstat -i# -s`	Section 4.1.1.3, page 139
Port statistics	Status information about specific low-levels: receive and reassembly	`atmstat -i# -r` `atmstat -i# -rv`	Section 4.1.1.6, page 142,
	Transmit and fragmentation	`atmstat -i# -t` `atmstat -i# -tv`	Section 4.1.1.5, page 140,
	SONET layer	`atmstat -i-S#` `atmstat -i# -Sv`	Section 4.1.1.7, page 143
	Complete listing of low-level status information (transmit, receive, and SONET)	`atmstat -i# -a` `atmstat -i# -av`	Section 4.1.1.8, page 145
	Constantly updated complete listing of low-level status information (transmit, receive, and SONET). Display can be set to provide ongoing incremental statistics, 1 second totals (deltas), or start-from-now counts.	`atmstat -i# -C`	
	Local port's ATM address	`ifatmconfig atm#`	Section 4.1.1.10, page 146
Driver information	IRIS ATM driver statistics	`atmstat -i# -d`	Section 4.1.3, page 148
VC information	Currently active VCs (PVCs and SVCs), including those used by `atmsigd` and `atmilmid` for protocol overhead purposes	`atmstat -i# -V`	Section 4.1.2.1, page 147

	Information	Command	More Info
LIS information	Local ATM address for IP logical network interface	`ifatmconfig atm#`	Section 4.1.4.1, page 150
	IP-to-PVC address resolution table, same information with ATM addresses	`atmarp -a` `atmarp -al`	Section 4.1.2.2, page 147
	IP-over-SVC information for each LIS (ATMARP server, transmit rate, and timeout value)	`ifatmconfig atm#`	Section 4.1.4.2, page 150

4.1.1 Displaying Board Information

You can display the following information about the board:

- Board configuration information (see Section 4.1.1.1, page 137)
- The firmware version (see Section 4.1.1.2)
- The board's current state (see Section 4.1.1.3)
- Transmission rates on board queues (see Section 4.1.1.4)
- Receive and reassembly status information (see Section 4.1.1.5)
- Transmit and fragmentation status information (see Section 4.1.1.6)
- SONET layer status information (see Section 4.1.1.7)
- All status information (see Section 4.1.1.8)
- A port's MAC address (see Section 4.1.1.9)
- A port's ATM address (see Section 4.1.1.10)

The following sections describe how to access this information.

4.1.1.1 Displaying Board Configuration Information

To display the settings of the board's operational parameters (that is, its configuration), use the command line below. Table 21, page 138, describes each of the parameters.

```
% /usr/etc/atmconfig -i# -s
```

The # variable identifies the port number.

Table 21. Board Configuration Parameters

Field	Description
sign	ATM-OC3c board's signature
vers	ATM-OC3c board's / FLASH EEPROM's version
xtype	Transmission type: 1 =XT_UNKNOWN 2 =XT_STS3C, SONET STS-3c physical layer at 155.52 Mbps 3 =XT_DS3=3, DS3 physical layer at 44.736 Mbps 4 =XT_4B5B=4, 4B/5B encoding physical layer at 100 Mbps 5 =XT_8B10B, 8B/10B encoding physical layer at 155.52 Mbps
mtype	Media type: 1 =MT_UNKNOWN 2 =MT_COAX, Coax cable 3 =MT_SMF, Single-mode fiber 4 =MT_MMF, Multi-mode fiber 5 =MT_STP, Shielded twisted pair 6 =MT_UTP, Unshielded twisted pair
maxvpibits	Maximum number of bits that can be used for a VPI. Range of possible values is 0 to 8.
maxvcibits	Maximum number of bits that can be used by a VCI. Range of possible values is 0 to 16.
xmt_large_size	Size (in bytes) of large-sized transmit buffers.
xmt_large_bufs	Number of large-sized transmit buffers.
xmt_small_size	Size (in bytes) of small-sized transmit buffers.
xmt_small_bufs	Number of small-sized transmit buffers.
rcv_large_size	Size (in bytes) of large-sized receive buffers.
rcv_large_bufs	Number of large-sized receive buffers.
rcv_small_size	Size (in bytes) of small-sized receive buffers.
rcv_small_bufs	Number of small-sized receive buffers. This size buffer is only used for AAL3/4.

4.1.1.2 Displaying the Firmware Version

To display the version of the firmware that is currently loaded into the board's dynamic random access memory (DRAM), use the following command line:

```
% /usr/etc/atmconfig -i# -V
```

The # variable identifies the particular port's number.

The retrieved version was calculated originally with the formula below:

((((*yy*-92)*13+*m*)*32+*d*)*24+*hr*)*60+*minute*

In this formula, *yy* is the final two digits from the year when the version was created (for example, 93 or 94), *m* is the numerical month (1-12), *d* is the numerical day (1-31), and *hr* and *minute* are the time (0-24 for hour and 0-60 for minute).

4.1.1.3 Displaying the Current Board State

To display the ATM-OC3c board's current state, use the following command line. The board can be in any one of three states described in Table 22, page 139.

```
% /usr/etc/atmstat -i# -s
```

The # variable identifies the particular port's number.

Table 22. ATM-OC3c Board States

State	Description
DEAD	The board is not responding in any manner. It may not have power, it may be loose, or it may be dysfunctional.
PRE-INIT	The board has power, but has not been initialized.
DOWN	The board is initialized and can communicate with the host. The interface to the network (that is, the SONET components) are not operating, so no data is being transmitted or received.
UP	The board is operating.

4.1.1.4 Displaying Transmission Rates on Board Queues

To display the current rates for the eight rate queues on an ATM-OC3c board, use the following command lines. Figure 30, page 140, shows the display.

```
% /usr/etc/atmstat -i# -q
```

The # variable identifies the particular port's number.

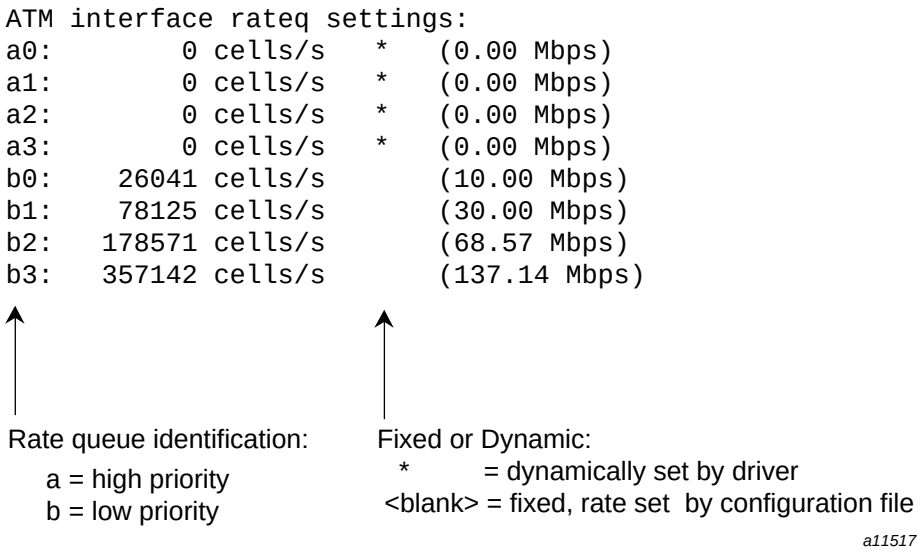


Figure 30. Rate Queue Information

The displayed rates indicate both the number of ATM cells per second and the number of bits of user data per second that are transmitted by that rate queue.

Note: Each ATM cell contains 48-bytes (384 bits) of user data.

4.1.1.5 Displaying Receive and Reassembly Status Information

To display information about the receive and reassembly functions, use the following command line:

```
% /usr/etc/atmstat -i# -r
```

The # variable identifies the particular port's number. Table 23, page 141, describes the displayed information.

Table 23. Receive Statistics: `atmstat -r`

Screen Display	Possible Values	Description
`/hw/atm/#: interface HW state:` *state*	# = 0 – 12 *state* = `UP`, `DOWN`, `INIT`, `DEAD`	# = port number *state* = current state of port
Receive protocol data unit (PDU) statistics:		
`PDUs received OK`	0 – *count*	Total PDUs received correctly.
`PDUs reassembly timeouts`	0 – *count*	PDU reassemblies that never completed.
`PDUs reassembly buffer size exceeded`	0 – *count*	PDU reassemblies that exceeded buffer size.
`PDUs AAL5 CRC-32 errors`	0 – *count*	PDUs that had AAL5 CRC errors.
`PDUs AAL5 length errors`	0 – *count*	PDUs that violated AAL5 size limit.
`PDUs unknown errors (none of the above)`	0 – *count*	PDUs that had errors that were not any of those listed above.
Receive Cell Statistics:		
`Cells OK`	0 – *count*	Total ATM cells received correctly.
`Cells dropped because invalid (unknown) VPI/VCI`	0 – *count*	ATM cells that were dropped due to unrecognized or bad VPI/VCI.
`Cells dropped due to no buffer available`	0 – *count*	ATM cells that were dropped due to lack of available (free) memory on the board.
`Cells unknown error (none of the above)`	0 – *count*	ATM cells that were dropped due to causes that are none of those listed above.
Receive SONET Errors:		
`SONET Section-layer BIP-8 errors`	0 – *count*	Section Overhead BIP-8 errors (B1 byte).
`SONET Line-layer BIP-24 errors`	0 – *count*	Line Overhead BIP-24 errors (that is, the BIP-8 [B2 byte] from the Line Overhead of each STS-1).
`SONET Path-layer BIP-8 errors`	0 – *count*	Path Overhead BIP-8 errors (B3 byte).

Screen Display	Possible Values	Description
`Correctable ATM HEC errors`	0 – *count*	Correctable errors that were detected by an ATM cell's header-error-control (HEC) field.
`Uncorrectable ATM HEC errors`	0 – *count*	Errors detected by an ATM cell's header-error-control (HEC) field that could not be corrected.

4.1.1.6 Displaying Transmit and Fragmentation Status Information

To display information about the transmit and fragmentation functions, use the following command line:

```
% /usr/etc/atmstat -i# -t
```

The # variable identifies the particular port's number. Table 24, page 142, describes the displayed information.

Table 24. Transmit Statistics: `atmstat -t`

Screen Display	Possible Values	Description
`/hw/atm/#: interface HW state:` *state*	# = 0 – 12 *state* = `UP`, `DOWN`, `INIT`, `DEAD`	# = port number *state* = current state of port
Transmit PDU Statistics:		
`PDUs transmitted OK`	0 – *count*	Total PDUs transmitted correctly.
`PDUs failed, unknown errors`	0 – *count*	All errors on transmitted PDUs.
Transmit Cell Statistics:		
`Cells ok`	0 – *count*	Total ATM cells transmitted correctly.
Transmit SONET (far-end) Errors:		

Screen Display	Possible Values	Description
`SONET Line-layer FEBEs`	0 – *count*	Line-layer (FEBE bits in Z2 byte) far-end-block-errors (FEBEs). This information is contained within received SONET frames, but it describes the error rate on the transmit data stream. Reported errors could have occurred anywhere along the entire line.
`SONET Path-layer FEBEs`	0 – *count*	Path-layer (FEBE bits in G1 byte) far-end-block-errors. This information is contained within received SONET frames but it describes the error rate on the transmit data stream. Reported errors could have occurred anywhere along the entire path.

4.1.1.7 Displaying SONET Layer Status Information

To display information about the SONET layer functions, use either of the following command lines:

```
% /usr/etc/atmstat -i# -S
% /usr/etc/atmstat -i# -Sv
```

The # variable identifies the particular port's number, and **`-v`** provides additional explanation about the meaning of each item of status information.

Table 25. SONET Statistics: `atmstat -Sv`

Screen Display	Possible Values	Description
`/hw/atm/#: interface HW state:` *state*	# = 0 – 12 *state* = `UP`, `DOWN`, `INIT`, `DEAD`	# = port's number *state* = current state of port

SONET Level Statistics:

Screen Display	Possible Values	Description
`Received Parity errors`	0 – *count*	Total of Section-layer BIP-8 errors (B1 byte), Path-layer BIP-8 errors (B3 byte), and Line-layer BIP-24 errors (that is, the BIP-8 [B2 byte] from the Line Overhead of each STS-1).
`Far End Bit Errors`	0 – *count*	Total of Line-layer (FEBE bits in Z2 byte) and Path-layer (FEBE bits in G1 byte) far-end-block-errors. This information is contained within received SONET frames but it describes the error rate on the transmit data stream. Reported errors could have occurred anywhere along the connection.
`Path Condition:` *state*	`OK` or `YELLOW`	`YELLOW` = Path layer yellow-signal error state currently exists in the receiving hardware.
`Path Alarm:` *state*	`OK` or `ALARM`	`ALARM` = Path layer alarm-indication-signal error state currently exists in the receiving hardware.
`Line Alarm:` *state*	`OK` or `ALARM`	`ALARM` = Line layer alarm-indication-signal state currently exists in the receiving hardware.
reg_contents SONET status bits:		Each SONET status item is one bit of an onboard register. *reg_contents* displays the value of that register. For each item: 0= not set; 1= set (event is current).
`Loss of Signal (LOS)`	0 or 1	Section layer: loss-of-signal error state currently exists in the receiving hardware.
`Loss of Frame (LOF)`	0 or 1	Section layer: loss-of-frame error state currently exists in the receiving hardware.
`Out of Frame (OOF)`	0 or 1	Section layer: out-of-frame error state currently exists in the receiving hardware.
`Far End Receive Failure (FERF)`	0 or 1	Line layer: far-end-receive-failure state currently exists in the receiving hardware.
`Line Alarm Indication Signal (LAIS)`	0 or 1	Line layer: alarm-indication-signal state currently exists in the receiving hardware.
`Loss of Path (LOP)`	0 or 1	Path layer: loss-of-pointer error state currently exists in the receiving hardware.

Screen Display	Possible Values	Description
`Path Alarm Indication Signal (PAIS)`	0 or 1	Path layer: alarm-indication-signal error state currently exists in the receiving hardware.
`Path Yellow Condition (Yel)`	0 or 1	Path layer: yellow-signal error state currently exists in the receiving hardware.
`Out Of Cell Delineation`	0 or 1	Out-of-cell-delineation state currently exists. The IRIS ATM receiving hardware is trying to synchronize with the cell boundaries in the SPE of the incoming SONET frame.
`Transmit Start of Cell error (TSOC)`	0 or 1	Start-of-cell error has occurred on the IRIS ATM transmit hardware.
`Transmit FIFO overrun`	0 or 1	FIFO overrun has occurred on the IRIS ATM transmit hardware.
`Receive FIFO overrun`	0 or 1	FIFO overrun has occurred on the IRIS ATM receive hardware.
`Receive FIFO underrun`	0 or 1	FIFO underrun has occurred on the IRIS ATM receive hardware.
mask `Path Signal Label (C2)`	any	Path layer: contents of the Signal Label (C2 byte) on incoming frames. A new value is captured when three consecutive frames have the same value. For the IRIS ATM hardware, this value should be 0x13 (hex) at all times.

4.1.1.8 Displaying All Status Information

To capture and display all the current status information (transmit, receive, and SONET), use the following command line:

```
% /usr/etc/atmstat -i# -av
```

The # variable identifies the port's number. Table 23, page 141, Table 24, page 142, and Table 25, page 143, describe the displayed information.

To display a constantly updating window of the current status information (transmit, receive, and SONET), use the following command line:

```
< open a UNIX shell/window that is at least 80x60 >
% /usr/etc/atmstat -i# -Cv
< d = start a new count every second, r = display accumulating total
since the last reboot, z = set all counters to 0 and
start accumulating new totals>
```

In this example, # identifies the port's number. Table 23, page 141, Table 24, page 142, and Table 25, page 143, describe the displayed information.

4.1.1.9 Displaying the Port's MAC Address

To display a medium access control (MAC) address, use the following command line:

```
% /usr/etc/atmconfig -i# -m
```

The # variable identifies the particular port's (board's) unit number.

4.1.1.10 Displaying the Port's ATM Address

To display a port's ATM address in hexadecimal format, invoke the `ifatmconfig` or `sigtest` utility, as follows:

- When the port is configured to support IP-over-ATM, use this command:

  `# ifatmconfig` *atm*#

  The *atm*# variable specifies one of the IP-over-ATM network interfaces associated with that port.

- When the port does not support IP-over-ATM, use this command:

```
# /usr/lib/atm/bin/sigtest
Menu selections:
 [0] Quit
 [1] Register to accept incoming calls
 [2] Attempt to setup a point-to-point call
 [3] Attempt to setup a point-to-multipoint call
Enter choice: 2
Using calling address: address type = NSAP
   Address = 47000580ffe1000000f21a01600800690422e900
<terminate sigtest by pressing the CTRL and C keys>
#
```

4.1.2 Displaying Virtual Channel Information

You can display the following information about the virtual channel:

- Currently active VCS (see Section 4.1.2.1, page 147)
- IP-to-VC address resolution table (see Section 4.1.2.2, page 147)

The following sections describe how to access this information.

4.1.2.1 Displaying Currently Active VCs

To display the table of currently active virtual circuits (VCs), use the following command line:

```
% /usr/etc/atmstat -i# -V
```

The # variable identifies the particular port's number.

The flags column in the terminal display indicates operational information about each active VC, which can be any combination of the flags described in Table 26, page 147:

Table 26. Active Virtual Channel Information

Flag	Description
READ	The channel is valid for reception.
WRITE	The channel is valid for transmission.
NOSNAP	The VC is not using LLC/SNAP encapsulation. This flag is only valid for PVCs.
IP	The VC is servicing an IP stack.

4.1.2.2 Displaying IP-to-VC Address Resolution Table

To display the contents of the IP-to-PVC address resolution table, use the following command line:

```
% /usr/etc/atmarp -a
```

To display known remote ATM addresses, as well as the other contents of the IP-to-PVC address resolution table, use the following command line:

```
% /usr/etc/atmarp -al
```

The flags column in the terminal display indicates status and information about each PVC, which can be any combination of the flags described in Table 27, page 148.

Table 27. ATMARP Address Resolution Table Flags

Flag	Description
CONN	The connection has been established for the VC.
COMPL	The ATM address for this IP address has been obtained.
NOSNAP	The VC is not using LLC/SNAP encapsulation. This flag is only valid for PVCs.
VALIDATE	The IP address has been validated with Inverse ARP.
PVC	The VC is a permanent virtual channel (PVC), not a switched one.
PEND	The connection has not yet been established; it is pending setup completion.
NAK	The ATMARP server has responded that it does not recognize this endpoint. The entry will soon be removed from the table.

4.1.3 Displaying ATM Driver Information

To display IRIS ATM driver statistics, use either of the following command lines. Table 28, page 149, describes the parameters displayed.

```
% /usr/etc/atmstat -i# -d
% /usr/etc/atmstat -i# -dv
```

The # variable identifies the particular port's number, and `-v` provides additional driver statistics.

Table 28. Driver Statistics: `atmstat -dv`

Driver Statistic	Description
`/hw/atm/#: interface HW state`	Specifies the indicated board's state. States are described in Table 22, page 139.
`Input packets`	Count of incoming ATM packets.
`Input bytes`	Count of total incoming bytes.
`Input packet drops`	Count of packets that were dropped. The count includes packets dropped due to buffer overflows and unknown VCC addresses.
`Output packets`	Count of outgoing ATM packets.
`Output bytes`	Count of total outgoing bytes.
`Output errors`	Currently unused.
`xcmd_dly`	Count of commands that were delayed (not immediately placed on the command queue) due to heavy use of the command interface.
`xmit_dly`	Count of transmit commands that were delayed (not immediately placed on the command queue) due to heavy use of the command interface.
intrs	Count of host-to-board interrupts.
b2hs	Count of board-to-host interrupts.
`xmit_reqs`	Count of transmit requests.
`h2b_kicks`	Number of times host has reset the board.
`xmit_intrs`	Count of transmit interrupts.
`odone_intrs`	Count of transmit done messages sent by board to host. When this count equals the `xmit_reqs` count, all data on the transmit queues has been processed.
`recv_intrs`	Count of receive interrupts.
`fet_stat`	Number of times host has retrieved board status.

4.1.4 Displaying LIS Information

For each logical IP subnetwork (LIS), you can display the following information:

- The local ATM address
- Local LIS information

4.1.4.1 Displaying the Local ATM Address

To display the ATM address for an endpoint (that is, an IP logical network interface) that is using IP-over-SVC, use the following command line:

```
% /usr/etc/ifatmconfig atm#
```

The # variable identifies the IP network interface (for example, `atm0` or `atm1`).

4.1.4.2 Displaying Local LIS Information

To display local information about a logical IP subnetwork (LIS), use the following command line:

```
% /usr/etc/ifatmconfig atm#
```

The # variable identifies the local endpoint (that is, the logical IP network interface) for the LIS.

4.1.5 Displaying PVC Information

For information about displaying PVC information, see Section 4.1.2, page 147.

4.2 Reading the Contents of ATM MIBs

The IRIS ATM management information database (MIB) is viewable with any simple network management protocol (SNMP) MIB browser, for example, the browsers included in Silicon Graphics' NetVisualyzer and IRIXPro applications. Complete instructions for using these applications are provided in the user documentation for these products (for example, in the *NetVisualyzer User's Guide*).

The path within the SNMP containment tree for the ATM UNI MIB is the following:

- By name:

```
iso.organization.dod.internet.private.enterprises.atmForum.atmForumUni
```

- By identification number:

```
1.3.6.1.4.1.353.2
```

Troubleshooting and Error Messages [5]

This chapter is a reference section containing a section that lists symptoms of common problems and a section with an alphabetical list of all the error messages that can be displayed by the IRIS ATM drivers and utilities.

5.1 Symptoms

The following sections describe symptoms of common problems.

5.1.1 `sigtest` Fails with Cause 47

When the `sigtest` utility fails with cause 47, it is probable that the adjacent switch does not support the traffic parameters you have selected. For example, not all switches support 3 BLLI selections.

5.1.2 MIB Browser Does Not Work

When the SNMP browser application does not offer the "Enterprise" or "atmForum" variables for viewing, the ATM MIB definition file is probably missing. When the application indicates it cannot create the MIB tree, the MIB2 definition file is probably missing. Use the following instructions to resolve the problem.

On the system running the SNMP browser application, verify that the ATM MIB and MIB2 definition files (`atmf_ilmi.mib` and `mib2`) are installed in the appropriate directory for the MIB viewing application that you are using. When IRIS ATM is installed with `inst`, the ATM MIB file is automatically placed in the `/usr/lib/netvis/mibs` directory on the system running the IRIS ATM software. You can copy the definition file from the IRIS ATM station to the system running the browser application. The appropriate location for all MIB definition files for the NetVisualyzer and IRIXPro Browser applications is the `/usr/lib/netvis/mibs` directory. To perform this verification step for the NetVisualyzer or IRIXPro Browser applications, use the following command:

```
% ls /usr/lib/netvis/mibs
atmf_ilmi.mib
mib2
```

If the files are not listed (that is, they do not exist), copy them to this directory from another workstation. If the files exist, verify that the `atmilmid` command is registered in the browser application system's `/etc/snmpd.remote.conf` file as a subagent to the main SNMP agent. The `atm-ilmi` entry in the `snmpd.remote.conf` file should be similar to the following one:

```
% grep atm-ilmi /etc/snmpd.remote.conf
1.3.6.1.4.1.353.2 IPaddress  23849  4  atm-ilmi
```

Note: This verification step must be done on the system where the SNMP viewing application (browser) is running. The *IPaddress* variable must identify a logical network interface on the system where the IRIS ATM ILMI software is running.

If this line exists, verify that the UDP socket (port) indicated in the `snmpd.remote.conf` entry matches the one specified in the `/etc/config/atmilmid.options` file, as explained in Section 3.7.2, page 128, and that *IPaddress* identifies the IRIS ATM logical network interface for the UNI in question.

If everything seems fine, kill and restart `snmpd`. If the problem persists, reinstall the IRIS ATM software and, if necessary, copy the ATM MIB file again to the system running the browser application. If the problem still persists, there is probably a problem with the SNMP MIB viewing application. Contact your support person for that application.

5.2 Error Messages

The following sections give an overview of ATM error messages and then lists them in alphabetical order.

5.2.1 Overview

The following section provides a listing of error messages and a brief discussion of the problems the message may indicate. The list contains only messages that indicate an error or problem; it does not contain informational messages that occur during normal operation.

Messages are alphabetized according to the following rules:

- Each message is alphabetized by the numerals (0–9) and letters (a–z) of the message's text. Numerals precede letters. Capitalization makes no difference. (Figure 31, page 155, shows the text of an error message.)
- Nonletters (for example, - or %) and blank spaces are shown in the text of the message, but are ignored in alphabetization. For example, the message `sm_open` is alphabetized as follows: `smnet`, `sm_open`, `smp`.
- When an error message includes an item that the software fills in for each instance of the message, this item is displayed in italic font and labeled with a generic name (for example, *filename*). The generic names are skipped for alphabetization purposes. For example, the error message `goofy not responding` is alphabetized among the "n" listings as *hostname* `not responding`. Common generic names used in this listing include *hostname*, *interfacename*, *version#*, *userentry*, *systemmessage*, *digit*, *filename*, and *hexnumeral*.

  Note: If you cannot find an error message in the listing, identify potential fill-in words, then look up the message without those words.
- The creator of each message is listed, in angled brackets, below the text of the message: (<*creator*>).

IRIS ATM error messages are written into the `/var/adm/SYSLOG` file or displayed at the terminal; some messages appear in both places. Within the `SYSLOG` file, each message is preceded by the date, time, host name, the name of the process that created the message, and its process ID number, as shown in the following figure. Only the text of the error message is included in the alphabetic list that follows.

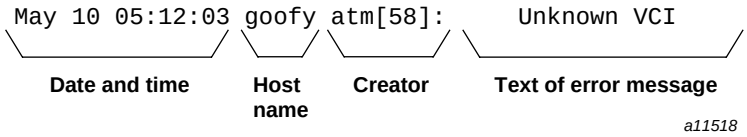


Figure 31. Error Message Format in `/var/adm/SYSLOG` File

Note: The list of error messages in this chapter covers only those unique to IRIS ATM. Standard system error messages, even when caused by the IRIS ATM code, are not covered.

5.2.2 Alphabetical List of Error Messages

This section lists ATM error messages in alphabetical order.

`Aborting config_up().`
`<signaling software>`

During an attempt to start up (build its stack), the signaling software encountered a problem and aborted the build. When this message follows a message indicating that a device file could not be opened or a PVC could not be bound, this means that the error occurred during initial startup of the IRIS ATM subsystem (for example, `atmconfig -u`).

`Accept failed:` *systemmessage*
`<sigtest>`

As requested by `sigtest`, the driver attempted to accept an incoming VC request, but the accept failed. That is, the `ioctl()` command `ATMIOC_ACCEPT` failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Bad address	EFAULT: An error occurred when the ATM software attempted to read the call's argument.
Interrupted system call	EINTR: While waiting for the accept call to complete from over the network, the driver was interrupted unexpectedly.
Invalid argument	EINVAL: The file descriptor was already bound, an internal value was invalid, or the incoming VC request queue was empty.
No space left on device	ENOSPC: The driver was not able to allocate an internal identifier for the SVC. This may indicate that too many VCs are already open.
No such device	ENODEV: The board was not in the UP or DOWN state. Or, the port was not operational.
Socket is not connected	ENOTCONN: The connection request is no longer valid. It has timed out, or been released by the calling party.

`AddParty ioctl failed:` *systemmessage*
`<sigtest>`

The driver was unable to add a party to the multipoint VC, as requested by `sigtest`. The `ioctl()` command `ATMIOC_ADDPARTY` failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the man page for `intro`(2):

Bad address	EFAULT: An error occurred when the ATM software attempted to read the call's argument.
Invalid argument	EINVAL: The SVC associated with the file descriptor is not connected or is not a multipoint connection `ATMIOC_MPSETUP` has not been called or did not succeed.
I/O error	EIO: The add party call was rejected by the network (an intermediate system) or by the called party. The reason (cause) for the rejection is provided in another error message.
No such device	ENODEV: The board was not in the UP or DOWN state. Or, the port was not operational.

`Address length must be between 1 and 15`
`<sigtest>`
The native-E.164 address that was entered does not have a valid count of digits. A digit is any of the following characters: 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9. The address must contain at least 1 digit and no more than 15.

`Address length (`*length*`) unacceptable, must be 40.`
`<sigtest>`
The ATM NSAP address that was entered contained an invalid count (*length* of user's entry) of hexadecimal characters. The address must be exactly 40 hexadecimal characters.

`Address string contains non-hexadecimal characters.`
`<sigtest>`
The ATM NSAP address that was entered contained an invalid character. Valid hexadecimal characters are the following: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, A, b, B, c, C, d, D, e, E, f, and F.

`atmarp: Bad port # in line #`*number* `in file` *filename*
`<atmarp>`
As the contents of the file specified on the `atmarp -f` command line were parsed, an invalid board unit (*unit*) was encountered.

Change the entry at the line indicated (*linenumber*) to an installed IRIS ATM board. The `/sbin/hinv` command displays the unit numbers associated with the currently installed boards.

`atmarp: couldn't ATMIOC_CREATEPVC:` *systemmessage*
`<atmarp>`

When attempting to open a virtual circuit for one of the entries in the IP-to-PVC mapping file, `atmarp` was unable to create the channel. The `ioctl()` command `ATMIOC_CREATEPVC` failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Address already in use	EADDRINUSE: The VPI/VCI pair is already in use (bound) by an IP network interface, so nothing was done with the new (or duplicate) entry.
Bad address	EFAULT: An error occurred during a data copy of a table entry.
Invalid argument	EINVAL: The IP-to-ATM address resolution table has an entry that is not currently valid. For example, one of the named ATM network interfaces is not currently configured and enabled.
No space left on device	ENOSPC: The maximum number of receive VCs has been reached or all the board's transmit buffer space has been allocated.
No such device	ENODEV: The IP-to-ATM address resolution table has an entry that is not currently valid. For example, one of the boards (ports) is not available (for example, not in the UP state or not installed).

`atmarp: couldn't ATMIOC_GETARPTAB:` *systemmessage*
`<atmarp>`

When attempting to display the current IP-to-ATM address resolution table, `atmarp` was unable to complete the task. The `ioctl()` command `ATMIOC_GETARPTAB` failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Bad address	EFAULT: An error occurred as the data was being copied. Try again.

Invalid argument	EINVAL: This message indicates a problem with the IRIS ATM software. Use `ifconfig` to disable the IRIS ATM network interfaces, use `atmconfig` to put the board into the DOWN state, use `inst` to remove and reinstall the IRIS ATM software, use `autoconfig` to build the driver into the operating system, then reboot the system.

`atmarp: couldn't ATMIOC_SETARP:` *systemmessage*
`<atmarp>`

When attempting to open a virtual circuit for one of the entries in the IP-to-ATM mapping file, `atmarp` was unable to map an IP address to the VPI/VCI address for the VC. The `ioctl()` command `ATMIOC_SETARP` failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Address family not supported by protocol family	EAFNOSUPPORT: This indicates a problem with the IRIS ATM software. Remove and reinstall the software; then use `autoconfig` and `reboot` to rebuild and start using a new operating system.
Bad address	EFAULT: An error occurred during a data copy.
Can't assign requested address	EADDRNOTAVAIL: The IP-to-ATM address resolution table is completely full. The file being loaded has more than 256 entries.
Invalid argument	EINVAL: This message indicates a problem with the IRIS ATM software.

`atmarp: couldn't open ATM device:` *systemmessage*
`<atmarp>`

The device file (for example, `/hw/atm/2`) for a hardware unit (for example, port 2) mentioned in the IP-to-PVC address resolution (AR) table could not be opened. This may indicate that the board was not located during the last power on, that the board is not active (UP), or that the IRIS ATM software was not installed correctly. The *systemmessage* (a standard system errno) provided in the message indicates the problem. Some system messages are described below; others are described in the man page for the `open()` system call.

No such device	ENODEV: An IRIS ATM board (port) mentioned in the AR table is not in an active state.

No such device or address	ENXIO: The hardware unit (#) specified on the command line (with `-i`) does not have a device file in the `/hw/atm` directory.
Permission denied	EACCESS: You must be super user (root).

To remedy ENODEV or ENXIO, do either of the following:

1. Remove the device from the mapping file.

   Use the command line below to list all the available IRIS ATM hardware units, then edit the `/usr/etc/pvc.conf` file so that it mentions only listed units (ports), and finally, invoke the `network.atm` script (or `atmarp -f`) to load a new address resolution table.

   ```
   # hinv | grep ATM
   ATM OC-3c unit 0: slot x, adapter x
   ATM OC-3c unit 1: slot x, adapter x
   ```

2. Make the device available.

   Invoke `hinv` to display the hardware devices that are recognized by the operating system. Then follow the appropriate step below:

   - If no IRIS ATM devices are listed, follow the verification instructions in Chapter 2, page 53, to verify that the IRIS ATM hardware and software have been installed correctly. Then, use `autoconfig` to build the IRIS ATM driver into the operating system, and reboot to start running the new operating system.
   - If at least one IRIS ATM device is listed, but others are not, reinstall unlisted IRIS ATM hardware making sure to set the unit jumper sets correctly or to configure the product so that the software assigns the unit numbers.
   - If all the IRIS ATM devices are listed, use `atmstat` to verify the state of the boards. Each board must be in the UP state. If a board is not UP, follow the instructions in Table 19, page 132, to make the board active.

`atmarp: couldn't open` *filename* `for reading.`
`<atmarp>`

The file specified on the `atmarp -f` command line could not be opened for reading. Verify that the file allows read-access and that it is a simple ASCII text file.

`atmarp: Couldn't resolve hostname` *hostname*`, line` *#number* `in file` *filename*
`<atmarp>`

As the contents of the file specified on the `atmarp -f` command line were parsed, a name was encountered that could not be mapped to an IP address.

Add the name (*name*) to the network information database (for example, the local `/etc/hosts` file or the NIS server) or edit the entry on the line indicated so that it matches an entry from the database.

`atmarp_input: unimplemented op: 0x`*hexnumber*
`<driver>`

While attempting to process a received ATMARP packet, the driver encountered an unknown command. This indicates that the sender of the packet uses ARP commands that are not implemented in this version of the IRIS ATM driver. There is no problem with the IRIS ATM product.

`atmarp: Invalid VCI in line` *#number* `in file` *filename*
`<atmarp>`

As the contents of the file specified on the `atmarp -f` command line were parsed, an invalid entry was encountered in the VPI column.

Change the entry at the line indicated (*linenumber*) to a valid entry. Valid entries range from 0 to 65535 digital or 0x0000 to 0xFFFF hexadecimal, inclusive.

`atmarp: Invalid VPI in line` *#number* `in file` *filename*
`<atmarp>`

As the contents of the file specified on the `atmarp -f` command line were parsed, an invalid entry was encountered in the VPI column. Valid entries range from 0 to 255 digital or 0x00 to 0xFF hexadecimal.

Change the entry at the line indicated (*linenumber*) to a valid entry, inclusive.

`atmarp: Malformed line` *#number* `in file` *filename*`:`
problematic_entry
`<atmarp>`

As the contents of the file specified on the `atmarp -f` command line were parsed, an error was encountered at the line indicated.

Edit the file as explained in Section 3.5.5, page 109.

`atmarp: unknown flag(s):` *flag*`: in line` *number*`# in file` *filename*
`<atmarp>`

While reading the IP-over-PVC configuration file (`/var/atm/pvc.conf`), `atmarp` encountered an invalid entry (*flag*) in the flags position of the line specified (*number*#). To resolve this, edit the file as explained in Section 3.5.5.1, page 109.

`atmconfig: ATMIOC_EXEC:` *systemmessage*
`<atmconfig>`

While attempting to download new firmware, `atmconfig` was unable to start the EEPROM write ("burn") program. The `ioctl()` command `ATMIOC_EXEC` failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Bad address	EFAULT: An error occurred during a data copy. Try the command again.
No permission match	EPERM: The process must have super-user access privileges.
Timer expired	ETIME: The board is not responding in a timely manner. It may be asleep or dysfunctional.

`atmconfig: ATMIOC_SETCONF:` *systemmessage*
`<atmconfig>`

While attempting to reconfigure the board, an error occurred. The `ioctl()` command `ATMIOC_SETCONF` failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Bad address	EFAULT: An error occurred during a data copy. Try the command again.
No permissions match	EPERM: You must be super user (root).
Timer expired	ETIME: The board is not responding in a timely manner. It may be asleep or dysfunctional.

`atmconfig: couldn't open device /hw/atm/# :` *systemmessage*
`<atmconfig>`

When `atmconfig` attempted to open the device file for the hardware (`/hw/atm/#`), it did not find the file. This may indicate that the hardware unit was not located during the last power on, that the board is not active (UP), or that the IRIS ATM software was not installed correctly. The *systemmessage* (a standard system errno) indicates more exactly what the problem is; explanation for the *systemmessage* is available in the `intro`(2) and `open`(2)man pages.

`atmconfig:/hw/atm/# couldn't reconfigure`
`<atmconfig>`

While attempting to reconfigure the hardware with the indicated unit number (#), `atmconfig -X` or `atmconfig -R` was unable to complete the task. A prior error message should indicate more detail about the problem.

`atmconfig:/hw/atm/#: couldn't read firmware` *filename*
`<atmconfig>`

When starting the process of downloading new firmware from the specified file (*filename*) to the hardware with the indicated unit number (#), `atmconfig -l` could not read the file containing the new program for burning the EEPROM. A prior message should clarify the reason.

`atmconfig:/hw/atm/#: firmware is up to date.`
`<atmconfig>`

The firmware currently on the hardware is the correct version for compatibility with the driver that is currently running. No new firmware has been downloaded to the hardware with the indicated unit number (#). When installing new IRIS ATM software, you do not need to download new firmware. The driver does this automatically as it starts running, whenever it discovers that the firmware on the board is incompatible.

`atmconfig:/hw/atm/#: interface is already UP`
`<atmconfig>`

When trying to put the indicated hardware unit (#) into the UP state, `ifconfig -u` discovered that the board was already UP.

`atmconfig:/hw/atm/# interface must be DOWN to download`
`<atmconfig>`

Before starting to download new firmware to the hardware unit (#), `atmconfig -l` discovered that the device was not in the DOWN state. Use `atmconfig -d` to put to hardware into the DOWN state, before continuing. For more information on the use of the `-d` option, see Section 3.2.2.2, page 86, for Origin2000 or Onyx 2 systems, or Section 3.3.4.3, page 100, for Challenge or Onyx systems.

`atmconfig:/hw/atm/# interface must be DOWN to reconfigure`
`<atmconfig>`

Before starting to reconfigure the hardware unit (#), `atmconfig -X` or `atmconfig -R` discovered that the device was not in the DOWN state. Use `atmconfig -d` to reset and put the hardware into the DOWN state, before reconfiguring it. For more information on the use of the `-d` option, see Section

3.2.2.2, page 86, for Origin2000 or Onyx 2 systems, or Section 3.3.4.3, page 100, for Challenge or Onyx systems.

`atmconfig:/hw/atm/#: interface not in DOWN mode`
`<atmconfig>`

When trying to put the hardware unit (#) into the UP state, `ifconfig -u` discovered that the device was not in the DOWN state. Use `atmconfig -d` to put the hardware into the DOWN state before invoking this command again. For more information on the use of these options, see Section 3.2.2.2, page 86, for Origin2000 or Onyx 2 systems, or Section 3.3.4.3, page 100, for Challenge or Onyx systems.

`atmconfig:/hw/atm/#: trouble programming firmware`
`<atmconfig>`

While downloading the new firmware to hardware unit (#), `atmconfig -l` encountered a problem. A prior error message should clarify the reason.

`atmconfig -F: ATMIOC_SETCONF failed:` *`systemmessage`*
`<atmconfig>`

While attempting to configure the board from its configuration file (*filename*), `atmconfig` was unable to do so. The `ioctl()` command `ATMIOC_SETCONF` failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Bad address	EFAULT: An error occurred during a data copy. Try the command again.
No permissions match	EPERM: You must be super user (root).
Timer expired	ETIME: The board is not responding in a timely manner. It may be asleep or dysfunctional.

`atmconfig -F : bad buf config: size not multiple of 8: file` *`filename`*
`<atmconfig>`

While configuring the board from its configuration file (*filename*), `atmconfig` found that one of the sizes specified for buffers is not a multiple of 8. Edit the file and try the command again.

`atmconfig -F : bad buf config: small > large: file` *`filename`*
`<atmconfig>`

While configuring the board from its configuration file (*filename*), `atmconfig` found that the size specified for the small buffers is larger than that specified for the large buffers. Edit the file and try the command again.

`atmconfig -F : bad buf config: total space is too large: file` *filename*
`<atmconfig>`
While configuring the board from its configuration file (*filename*), `atmconfig` found that the sum total of all the specified buffer sizes is more than the board can accommodate (which is 1,966,080 bytes). Edit the file and try the command again.

`atmconfig -F : bad` *buffers* `declaration: line` *number* `file` *filename*
`<atmconfig>`
While configuring the board from its configuration file (*filename*), `atmconfig` found that the indicated line (*number*) could not be parsed. The problematic entry is for the indicated *buffers*: `XLBUF`=large transmit, `XSBUF`=small transmit, `RLBUF`= large receive, or `RSBUF`=small receive buffers. Edit the file and try the command again.

`atmconfig -F : bad rate value in line` *number* `file` *filename*
`<atmconfig>`
While attempting to reconfigure the rate queues on the IRIS ATM board from the configuration file (*filename*), the command encountered an unrecognized character in the position where it expected to read the rate queue's rate. Edit the file, as explained in Section 3.3.3, page 93, then invoke the command again.

`atmconfig -F : couldn't ATMIOC_SETRATEQ: line # file` *filename*
`:` *systemmessage*
`<atmconfig>`
While attempting to reconfigure the rate queues on the IRIS ATM board from the configuration file (*filename*), the command's `ioctl()` call to the board (`ATMIOC_SETRATEQ`) failed with the error indicated in the *systemmessage* (a standard system errno); additional explanation for the *systemmessage* is available in the `intro`(2) man page. Verify that the rate specified on the problematic line in the file is supported by the hardware.

`atmconfig -F : couldn't get original configuration:` *systemmessage*`''`
`<atmconfig>`
While configuring the board from its configuration file, `atmconfig` found that the communication path to the board is not working. Specifically, the `ATMIOC_GETCONF ioctl()` call failed. This probably indicates that the board is not responding. The *systemmessage* (a standard system errno) for the failure

indicates what the problem is, as explained below. Additional explanation for the *systemmessage* is available in the `intro`(2) man page.

Bad address	EFAULT: An error occurred during a data copy. Try the command again.
No such device	ENODEV: The IRIS ATM board (port) is not installed.
Timer expired	ETIME: The board is not responding in a timely manner. It may be asleep or dysfunctional.

`atmconfig -F : couldn't open input file:` *filename*
`:` *systemmessage*
`config_file );`
`<atmconfig>`

While attempting to reconfigure the IRIS ATM board, the command could not open the indicated configuration file (*filename*) for reading. It could be that the file does not exist or that its access modes do not allow reading. The *systemmessage* (a standard system errno) indicates more exactly what the problem is; additional explanation for the *systemmessage* is available in the `intro`(2)man page.

`atmconfig -F : interface must be UP or DOWN.`
`<atmconfig>;`

While attempting to reconfigure the IRIS ATM board, the command discovered that the board is not in the UP or DOWN state. Invoke `atmconfig -u` to bring the board UP. If this fails, use `atmconfig -d` followed by `atmconfig -u`. If this also fails, invoke `atmconfig -r`, `atmconfig -d`, and `atmconfig -u`. If this does not work, power cycle the machine. For more information on the use of these options, see Section 3.2.2.2, page 86, for Origin2000 or Onyx 2 systems, or Section 3.3.4.3, page 100, for Challenge or Onyx systems.

`atmconfig -F : must be in DOWN mode to change buffer configuration:`
`line` *number* `file` *filename*
`<atmconfig>`

While configuring the board's buffer sizes from the configuration file, `atmconfig` found that the board was not in the correct state for reconfiguration. The board must be in the DOWN state. Use the `atmconfig -d` command to change the board's state, then invoke the command again. For more information on the use of the `-d` option, see Section 3.2.2.2, page 86, for Origin2000 or Onyx 2 systems, or Section 3.3.4.3, page 100, for Challenge or Onyx systems.

`atmconfig -F : syntax error line` *linenumber* `file` *filename*
`<atmconfig>`

While attempting to reconfigure the IRIS ATM board, the command could not interpret something on the indicated line (*linenumber*). Edit the file, as explained in Section 3.3.3, page 93, then invoke the command again.

`atmconfig -F : unknown rate queue designation in line # file` *filename*
`<atmconfig>`

While attempting to reconfigure the rate queues on the IRIS ATM board from the configuration file (*filename*), the command encountered an unrecognized string in the position of the specified line where it expected to read the rate queue's identification value. Edit the file, as explained in Section 3.3.3, page 93, then invoke the command again.

`atmconfig: Invalid buffer configuration for card.`
`<atmconfig>`

While attempting to reconfigure the allocation of onboard memory, `atmconfig -X` or `atmconfig -R` discovered that the newly specified configuration is invalid. For example, the total byte count for all buffers (receive and transmit) may exceed the allowed maximum, or the specified size for large-sized buffers may be larger than the small-sized buffers, or one of the sizes may not be a multiple of 8.

`atmconfig: lflag: trouble with ATMIOC_VERS:` *systemmessage*
`<atmconfig>`

When starting the process of downloading new firmware, `atmconfig -l` encountered a problem. The `ioctl()` command `ATMIOC_VERS` failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Bad address	EFAULT: An error occurred during a data copy. Try the command again.
Timer expired	ETIME: The board did not process the command and a relatively long period of time passed. The board may be very busy or may not be functional.

`ATMIOC_BINDVC:` *systemmessage*
`<atmtest>`

When attempting to set up a virtual circuit for the test, `atmtest` was unable to create the VC. The `ioctl()` command `ATMIOC_CREATEPVC` failed. The *systemmessage* (a standard system errno) indicates what the problem is, as

summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Address already in use	EADDRINUSE: The VPI/VCI pair is already in use (bound) by another VC, so nothing was done with the new (duplicate) entry.
Bad address	EFAULT: An error occurred during a data copy of a table entry.
Invalid argument	EINVAL: The file descriptor has incorrect read or write mode.
No space left on device	ENOSPC: The maximum number of receive VCs has been reached or all the board's transmit buffer space has been allocated.
No such device	ENODEV: The port associated with the `-i` option is not responding. For example, it is not in the UP state or not installed.

`ATMIOC_GETOPT:` *systemmessage*
`<atmtest>`

While attempting to retrieve the options on the board, `atmtest -O` encountered a problem. The `ioctl()` command `ATMIOC_GETOPT` failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Bad address	EFAULT: An error occurred during a data copy.
No permission match	EPERM: You must have super-user access privileges.
No such device	ENODEV: The board unit specified on the command line is not in the UP or DOWN state.
Timer expired	ETIME: The board is not responding in a timely manner. It may be very busy, asleep, or dysfunctional.

`ATMIOC_GETRATEQ:` *systemmessage*
`<atmstat>`

While attempting to retrieve the rate queue settings from the board, `atmconfig -q` encountered a problem. The `ioctl()` command `ATMIOC_GETRATEQ` failed. The *systemmessage* (a standard system errno) for the

failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Bad address	EFAULT: An error occurred during a data copy. Try again.
Interrupted system call	EINTR: The system call was interrupted and did not complete. Try again.
Not enough space	ENOMEM: The communication path between the driver and the board is currently busy. Try the command again.
No such device	ENODEV: The board is not in the UP state.

`ATMIOC_GETSTAT:` *systemmessage*
`<atmstat>`

While attempting to retrieve status information from the board, `atmstat` encountered a problem. The `ioctl()` command `ATMIOC_GETSTAT` failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Bad address	EFAULT: An error occurred during a data copy. Try the command again.
Interrupted system call	EINTR: The system call was interrupted and did not complete. Try again.
Not enough space	ENOMEM: The communication path between the driver and the board is currently busy. Try the command again.

`ATMIOC_GETVCTAB:` *systemmessage*
`<atmstat>`

While attempting to retrieve the virtual circuit table from the board, `atmconfig -V` encountered a problem. The `ioctl()` command `ATMIOC_GETVCTAB` failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Bad address	EFAULT: An error occurred during a data copy. Try again.
Invalid argument	EINVAL: This error indicates a problem or incompatibility with the IRIS ATM software.

No such device	ENODEV: The board is not in the UP state.

`ATMIOC_SETOPT:` *systemmessage*
`<atmtest>`

While attempting to set the options on the board, `atmtest -o` encountered a problem. The `ioctl()` command `ATMIOC_SETOPT` failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

No permissions match	EPERM: You must be super user (root).
No such device	ENODEV: The board unit specified on the command line is not in the UP or DOWN state.
Timer expired	ETIME: The board is not responding in a timely manner. It may be very busy, asleep, or dysfunctional.

`atm_set_rateqs: failed to set rq` *number*`: error =` *errornumber*
`<driver>`

While attempting to reconfigure the rate for the indicated rate queue (number), the driver encountered an error. The error (*errornumber*) supplied by the board indicates the problem.

`ATMIOC_VERS:` *systemmessage*
`<atmtest>`

While attempting to retrieve the firmware version from the board, `atmtest -V` encountered a problem. The `ioctl()` command `ATMIOC_VERS` failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Bad address	EFAULT: An error occurred during a data copy. Try the command again.
No such device	ENODEV: The board unit specified on the command line is not in the UP or DOWN state.
Timer expired	ETIME: The board did not process the command and a relatively long period of time passed. The board may be very busy or may not be functional.

`atmstatus: couldn't ATMIOC_GETOPT:` *systemmessage*
`<atmstat>`

While attempting to retrieve the options on the board, `atmstat -o` encountered a problem. The `ioctl()` command `ATMIOC_GETOPT` failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Bad address	EFAULT: An error occurred during a data copy.
No permission match	EPERM: The process must have super-user access privileges.
No such device	ENODEV: An IRIS ATM board (port) with a unit number specified on the command line is not in an active state (not DOWN or UP).
Timer expired	ETIME: The board is not responding in a timely manner. It may be asleep or dysfunctional.

`ATM: svc_sigd_msg: Unknown putmsg opcode:` *number*`!`
`<signaling software>`

While attempting to place a message on the IRIS ATM signaling software's queue, an error occurred because the message is not a known (recognized) type. This indicates incompatible IRIS ATM software modules. Use the `inst` utility to remove and reinstall all the IRIS ATM software.

`atmtest: bad IP address given:` *IPaddress*
`<atmtest>`

The string specified with the `-B` option could not be resolved into an IP address. This could mean that the provided host name does not exist in the `/etc/hosts` file.

`atmtest: data integrity error; offset = 0x`*hexnumber*
`<atmtest>`

While performing the checksum on received data, `atmtest` discovered an error. The error is located at the indicated offset position.

`atmtest: data integrity error, offset = 0x`*hexnumeral*
`<atmtest>`

While comparing the incoming data to the transmitted data, `atmtest -Xrw -C` discovered a difference (that is, an error). The *hexnumeral* specifies the problematic buffer.

`atmtest: mpin() failed.`
systemmessage

`<atmtest>`

While attempting to allocate memory to handle the outgoing data, `atmtest`'s `mpin()` call failed. This indicates a problem with the operating system or the system. For example, the memory is totally occupied by other processes. Additional explanation for the *systemmessage* is available in the `intro`(2) man page. There is nothing wrong with the IRIS ATM software.

`atmtest: -N incompatible with reading`
`<atmtest>`

The `-N` option of `atmtest` expected an accompanying `-Xwo` or `-Xrw`. The `-N` option cannot be used for a VC opened for receiving (reading) only.

`atmtest -N: length must be a multiple of 48`
`<atmtest>`

The argument for `-N` must be a multiple of 48.

`atmtest: packet loss on /hw/atm/# vpi=0x`*hexnumber* `vci=0x`*hexnumber*
`<atmtest>`

While attempting to read the incoming data (that is, the data that was transmitted by `atmtest -Xw`), `atmtest -Xr` encountered an EINTR error caused by dropped (lost) packets on the VC and hardware unit (#) that are specified in the message. Use the `atmstat` command to discover the exact nature of the problem.

`atmtest: unknown -X parameters`
`<atmtest>`

The `-X` option of `atmtest` expects one of the following arguments: `ro` (read-only VC), `wo` (write-only VC), or `rw` (read-write VC).

`can't open <`*filename*
`<atmconfig>`

While attempting to download new firmware, `atmconfig -l` could not open the indicated file for reading. To resolve this, reinstall the IRIS ATM software.

`can't read file header from <`*filename*`>`
`<atmconfig>`

While attempting to download new firmware, `atmconfig -l` could not read the header on the indicated file. To resolve this, reinstall the IRIS ATM software.

`can't read section` *number* `header in <`*filename*`>`
`<atmconfig>`

While attempting to download new firmware, `atmconfig -l` found that it could not read the indicated section (*number*) of the file containing the firmware. To resolve this, reinstall the IRIS ATM software.

`can't skip a.out header in <`*filename*`>`
`<atmconfig>`
While attempting to download new firmware, `atmconfig -l` could not skip the compiler's header on the indicated file containing executable firmware. To resolve this, reinstall the IRIS ATM software.

`Cause of failure =` *number* `(`*cause_message*`)`
`<sigtest>`
The setup call was rejected at the other endpoint or along the ATM network for the cause specified in the error message. ATM rejection causes are summarized in Table 29, page 203, and Table 30, page 206.

`Could not open the file:` *filename*
`<browser>`
On the system running the SNMP browser application, verify that the ATM MIB and the standard MIB2 definition files (`atmf_ilmi.mib` and `mib2`) are installed in the appropriate directory for the MIB viewing application that you are using. When IRIS ATM is installed with `inst`, the ATM MIB file is automatically placed in the `/usr/lib/netvis/mibs` directory on the system running the IRIS ATM software. You can copy the definition file from the IRIS ATM station to the system running the browser application. The appropriate location for all MIB definition files for the NetVisualyzer and IRIXPro Browser applications is the `/usr/lib/netvis/mibs` directory. To perform this verification for the NetVisualyzer or IRIXPro applications, use the command below:

```
% ls /usr/lib/netvis/mibs
atmf_ilmi.mib
mib2
```

If the files are not listed (that is, it do not exist), copy them to the directory. If the files exist, verify that `atmilmid` is registered in the browser application system's `/etc/snmpd.remote.conf` file as a subagent to the main SNMP agent. The `atm-ilmi` entry in the `snmpd.remote.conf` file should be similar to the one shown below:

```
% grep atm-ilmi /etc/snmpd.remote.conf
1.3.6.1.4.1.353.2 IPaddress  23849  4  atm-ilmi
```

Note: This verification step must be done on the system where the SNMP viewing application (browser) is running. The *IPaddress* must identify the logical network interface on the system where the IRIS ATM ILMI software is running.

If this line exists, verify that the UDP socket (port) indicated in the `snmpd.remote.conf` entry matches the one specified in the `/etc/config/atmilmid.options` file, as explained in Section 3.7.2, page 128, and that the *IPaddress* identifies the IRIS ATM logical network interface for the UNI in question.

If everything seems fine, kill and restart `snmpd`. If the problem persists, reinstall the IRIS ATM software and, if necessary, copy the ATM MIB file again to the system running the browser application. If the problem still persists, there is probably a problem with the SNMP MIB viewing application. Contact your support person for that application.

`couldn't ATMIOC_GETSTAT:` *systemmessage*
`<atmconfig>`

When `atmconfig` attempted to retrieve the current status for the board, an error occurred. The `ioctl()` command `ATMIOC_GETSTAT` failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Bad address	EFAULT: An error occurred during a data copy. Try the command again.
Interrupted system call	EINTR: The system call was interrupted and did not complete. Try again.
Not enough space	ENOMEM: The communication path between the driver and the board is currently busy. Try the command again.

`couldn't ATMIOC_SETOPT:` *systemmessage*
`<atmconfig>`

While attempting to reconfigure the board with new operational options, `atmconfig -o` encountered a problem. The `ioctl()` command `ATMIOC_SETOPT` failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

No permissions match	EPERM: You must be super user (root).

No such device	ENODEV: The board unit specified on the command line is not in the UP or DOWN state.
Timer expired	ETIME: The board did not process the command and a relatively long period of time passed. The board may be very busy or may not be functional.

`couldn't ATMIOC_SETRATEQ:` *systemmessage*
`<atmconfig>`

When attempting to open a virtual circuit for one of the entries in the IP-to-ATM mapping file, `atmconfig -Q` was unable to configure a timer (rate queue). The `ioctl()` command `ATMIOC_SETRATEQ` failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Bad address	EFAULT: An error occurred during a data copy. Try again.
Interrupted system call	EINTR: The system call was interrupted and did not complete. Try again.
No such device	ENODEV: One of the boards mentioned in the IP-to-ATM mapping file is not in the UP state.
Not enough space	ENOMEM: The communication path between the driver and the board is currently busy. Try the command again.

`couldn't bind LIS to port:` *systemmessage*
`<ifatmconfig>`

When `ifatmconfig` was attempting to bind the LIS (that is, the logical IP network interface) to the port specified on the command line, it encountered a problem. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Device or resource busy	EBUSY: The logical network interface identified on the command line is already configured and enabled. Use `ifconfig down` to disable it before trying again.

Invalid argument	EINVAL: The port unit number specified on the command line is not valid. Use `hinv` to verify the unit number before trying again.

`couldn't get arpserver:` *systemmessage*
`<ifatmconfig>`

While attempting to display information about an LIS, the `ifatmconfig` command could not retrieve the address of the ATMARP server. The *systemmessage* (a standard system errno) for the failure indicates what the problem is. Additional explanation is available in the `intro`(2) man page.

`couldn't get atm address:` *systemmessage*
`<ifatmconfig>`

While attempting to display information about an LIS, the `ifatmconfig` command could not retrieve the ATM address assigned to the LIS. The `ATMIOC_GETATMADDR ioctl()` call failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Bad address	EFAULT: An error occurred when the IRIS ATM driver attempted to return the retrieved information.
No such device	ENODEV: The board was not in the UP or DOWN state. Or, the port was not operational.

`couldn't get configuration:` *systemmessage*
`<atmconfig>`

While attempting to reconfigure the allocation of onboard memory, `atmconfig -X` or `atmconfig -R`, could not retrieve the current configuration from the board because the `ATMIOC_GETCONF ioctl()` call failed. This may indicate that the board is not responding. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as explained below. Additional explanation for the *systemmessage* is available in the `intro`(2) man page.

Bad address	EFAULT: An error occurred during a data copy. Try the command again.
No such device	ENODEV: An IRIS ATM board (port) with a unit number mentioned in the AR table is not in an active state.

Timer expired	ETIME: The board is not responding in a timely manner. It may be asleep or dysfunctional.

`couldn't get default rate:` *systemmessage*
`<ifatmconfig>`
While attempting to display information about an LIS, the `ifatmconfig` command could not retrieve the cell rate used for traffic to that LIS. The *systemmessage* (a standard system errno) for the failure indicates what the problem is. Additional explanation is available in the `intro`(2) man page.

`couldn't get port #:` *systemmessage*
`<ifatmconfig>`
While attempting to display information about an LIS, the `ifatmconfig` command could not retrieve the port assigned to the LIS. The *systemmessage* (a standard system errno) for the failure indicates what the problem is. Additional explanation is available in the `intro`(2) man page.

`couldn't get vcc time-out:` *systemmessage*
`<ifatmconfig>`
While attempting to display information about an LIS, the `ifatmconfig` command could not retrieve the timeout for inactive VCs to that LIS. The *systemmessage* (a standard system errno) for the failure indicates what the problem is. Additional explanation is available in the `intro`(2) man page.

`couldn't get VCC time-out:` *systemmessage*
`<ifatmconfig>`
After assigning the requested VC timeout for an LIS, the driver was unable to read it back. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

`couldn't ioctl:` *systemmessage*
`<atmconfig>`
While trying to retrieve the current configuration from the board, `atmconfig -s` encountered an error. The `ioctl()` command `ATMIOC_GETCONF` failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Bad address	EFAULT: An error occurred during a data copy. Try the command again.

Timer expired	ETIME: The board did not process the command and a relatively long period of time passed. The board may be very busy or may not be functional.

`couldn't ioctl( ATMIOC_GETIOSTAT ):` *systemmessage*
`<atmstat>`

While attempting to retrieve I/O driver statistics, `atmconfig -d` encountered a problem. The `ioctl()` command `ATMIOC_GETIOSTAT` failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2)man page. For more information on the use of the `-d` option, see Section 3.2.2.2, page 86, for Origin2000 or Onyx 2 systems, or Section 3.3.4.3, page 100, for Challenge or Onyx systems.

Bad address	EFAULT: An error occurred during a data copy. Try again.

`couldn't open device /hw/atm/#:` *systemmessage*
`<atmstat>`

When `atmstat` attempted to open the device file for the indicated hardware unit (#), it did not find the file. This may indicate that the board was not located during the last power on, that the board is not active (UP), or that the IRIS ATM software was not installed correctly. The *systemmessage* (a standard system errno) provided in the message indicates the problem. Some system messages are described below; others are described in the man page for the `open`(2) system call.

No such device	ENODEV: An IRIS ATM hardware (port) with the unit number indicated (#) is not in an active (UP) state.
No such device or address	ENXIO: The IRIS ATM board (port) specified on the command line (with `-i`) does not have a device file in the `/hw/atm` directory.
Permission denied	EACCESS: You must be super user (root).

`couldn't open:` *systemmessage*
`<atmtest>`

When `atmtest` attempted to open the device file, it failed. This may indicate that the board was not located during the last power on, that the board is not active (UP), or that the IRIS ATM software was not installed correctly. The *systemmessage* (a standard system errno) for the failure indicates what the

problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) and `open`(2) man pages.

No such device	ENODEV: An IRIS ATM hardware (port) with the unit number indicated (#) is not in an active (UP) state.
No such device or address	ENXIO: The IRIS ATM hardware (port) specified on the command line (with `-i`) does not have a device file in the `/hw/atm` directory.
Permission denied	EACCESS: You must be super user (root).

`couldn't read:` *systemmessage*
`<atmtest>`
While attempting to read the incoming data (that is, the same data that was transmitted by `atmtest -Xw`), `atmtest -Xr` encountered an error. The *systemmessage* (a standard errno) specifies the reason. Further explanation is available in the `intro`(2) and `read`(2) man pages.

`couldn't read:` *systemmessage*
<atmtest>
When `atmtest -Xro` attempted to read incoming data, it failed. The *systemmessage* (a standard errno) specifies the reason. Further explanation is available in the `intro`(2) and `read`(2)man pages.

`couldn't set ARP server: EBUSY`
`<ifatmconfig>`
When `ifatmconfig` was attempting to assign the ATM ARP server for the LIS (that is, the logical IP network interface), it found that the interface was already configured and enabled. Use `ifconfig down` to disable it before trying again.

`couldn't set rate:` *systemmessage*
`<ifatmconfig>`
The driver was unable to assign the requested VC transmission rate to this logical network interface. The *systemmessage* (a standard system errno) for the failure indicates what the problem is. Additional explanation is available in the `intro`(2) man page.

`couldn't set VCC time-out:` *systemmessage*
`<ifatmconfig>`

The driver was unable to assign the requested VC timeout for an LIS. The *systemmessage* (a standard system errno) for the failure indicates what the problem is. Additional explanation is available in the `intro`(2) man page.

`couldn't write:` *systemmessage*
`<atmtest>`
While attempting to write the transmit data to the ATM subsystem, `atmtest -Xw` encountered an error. The *systemmessage* (a standard errno) specifies the reason. Further explanation is available in the `intro`(2) and `write`(2) man pages.

`couldn't write:` *systemmessage*
`<atmtest>`
When `atmtest -Xwo` attempted to write outgoing data, it failed. The *systemmessage* (a standard errno) specifies the reason. Further explanation is available in the `intro`(2) and `write`(2) man pages.

`Download` *count* `words to address 0x08`*hexnumeral* `failed.`
`<atmconfig>`
The attempt to download the indicated number of words (*count*) to EEPROM address (*08hexnumeral*) did not succeed. The reason is provided in an adjacent error message.

`E.164 address must contain only decimal digits`
`<sigtest>`
The native-E.164 address that was entered contained invalid characters. Each entry must be a digit. A digit is any of the following characters: 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9.

`Error binding to VC on device /hw/atm/#:` *systemmessage*
`<signaling software>`
The signaling software was unable to open a PVC on the hardware unit (port) indicated (#). The `ATMIOC_CREATEPVC ioctl()` call to the IRIS ATM driver failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Address already in use	EADDRINUSE: The VPI/VCI pair is already in use (bound) by an IP network interface, so nothing was done with the new (or duplicate) entry.

Bad address	EFAULT: An error occurred during a data copy of a table entry.
Invalid argument	EINVAL: The IP-to-ATM address resolution table has an entry that is not currently valid. For example, one of the named ATM network interfaces is not currently configured and enabled.
No space left on device	ENOSPC: The maximum number of receive VCs has been reached or all the board's transmit buffer space has been allocated.
No such device	ENODEV: The IP-to-ATM address resolution table has an entry that is not currently valid. For example, one of the hardware units (ports) is not available (for example, not in the UP state or not installed).

`ERROR: Error binding to VC on device /hw/atm/#: (`*systemmessage*`) <signaling software>`

The signaling software was unable to open a PVC on the hardware unit indicated (#). The `ATMIOC_CREATEPVC ioctl()` call to the IRIS ATM driver failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Address already in use	EADDRINUSE: The VPI/VCI pair is already in use (bound) by an IP network interface, so nothing was done with the new (or duplicate) entry.
Bad address	EFAULT: An error occurred during a data copy of a table entry.
Invalid argument	EINVAL: The IP-to-ATM address resolution table has an entry that is not currently valid. For example, one of the named ATM network interfaces is not currently configured and enabled.
No space left on device	ENOSPC: The maximum number of receive VCs has been reached or all the board's transmit buffer space has been allocated.
No such device	ENODEV: The IP-to-ATM address resolution table has an entry that is not currently valid. For example, one of the hardware units (ports) is not

available (for example, not in the UP state or not installed).

`ERROR: Error binding to VC on device /hw/atm/# (`*systemmessage*`), aborting config_up()`
`<signaling software>`

During an attempt to start up and build its stack, the signaling software encountered a problem and aborted the build. The problem is that the signaling software was unable to open a PVC on the hardware unit indicated (#). The `ATMIOC_CREATEPVC ioctl()` call to the IRIS ATM driver failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Address already in use	EADDRINUSE: The VPI/VCI pair is already in use (bound) by an IP network interface, so nothing was done with the new (or duplicate) entry.
Bad address	EFAULT: An error occurred during a data copy of a table entry.
Invalid argument	EINVAL: The IP-to-ATM address resolution table has an entry that is not currently valid. For example, one of the named ATM network interfaces is not currently configured and enabled.
No space left on device	ENOSPC: The maximum number of receive VCs has been reached or all the board's transmit buffer space has been allocated.
No such device	ENODEV: The IP-to-ATM address resolution table has an entry that is not currently valid. For example, one of the hardware units (ports) is not available (for example, not in the UP state or not installed).

`ERROR: Error in atm read on /hw/atm/#:` *systemmessage*
`<signaling software>`

As the signaling software attempted to read incoming data on the indicated hardware unit (that is, device file `dev/atm#`) an error occurred, as described by the *systemmessage*. Additional explanation for the *systemmessage* is available in the `intro`(2) man page.

`ERROR: Error in atm write on /hw/atm/#:` *systemmessage*
`<signaling software>`

As the signaling software attempted to write outgoing data to the indicated hardware unit (that is, device file `/hw/atm/#`) an error occurred, as described by the *systemmessage*. Additional explanation for the *systemmessage* is available in the `intro`(2) man page.

`ERROR: Error mpin()ing output buffer for /hw/atm/# (`*systemmessage*`)`
`<signaling software>`

While attempting to allocate memory to handle the outgoing data, the signaling software's `mpin()` call failed. This indicates a problem with the operating system or the system. For example, the memory is totally occupied by other processes. Additional explanation for the *systemmessage* is available in the `intro`(2) man page. There is nothing wrong with the IRIS ATM software.

`ERROR: Error opening atm device /hw/atm/# (`*systemmessage*`)`
`<signaling software>`

The signaling software was unable to open a device file for the hardware unit indicated (#). The reason for the failure is provided in the *systemmessage*, for which additional explanation is available in the `intro`(2) man page.

`ERROR: Error opening atm device /hw/atm/# (`*systemmessage*`),`
`aborting config_up()`
`<signaling software>`

During an attempt to start up and build its stack, the signaling software encountered a problem and aborted the build. The problem is that the device file for the indicated hardware unit (#) could not be opened due to the error indicated by the *systemmessage*. Additional explanation for the *systemmessage* is available in the `intro`(2) man page.

`Error in atm read on /hw/atm/#:` *systemmessage*
`<signaling software>`

As the signaling software attempted to read incoming data on the indicated hardware unit (that is, device file `/hw/atm/#`) an error occurred, as described by the *systemmessage*. Additional explanation for the *systemmessage* is available in the `intro`(2) man page.

`Error (`*systemmessage*`) in atm write on device /hw/atm/#`
`<signaling software>`

As the signaling software attempted to write outgoing data to the indicated hardware unit (that is, device file `/hw/atm/#`) an error occurred, as described

by the *systemmessage*. Additional explanation for the *systemmessage* is available in the `intro`(2) man page.

`Error mpin()ing output buffer for /hw/atm/#:` *systemmessage*
`<signaling software>`

While attempting to allocate memory to handle the outgoing data, the signaling software's `mpin()` call failed. This indicates a problem with the operating system or the system. For example, the memory is totally occupied by other processes. Additional explanation for the *systemmessage* is available in the `intro`(2) man page. There is nothing wrong with the IRIS ATM software.

`Error opening atm device /hw/atm/#:` *systemmessage*
`<signaling software>`

The signaling software was unable to open a device file for the hardware unit indicated (that is, device file `/hw/atm/#`). The reason for the failure is provided in the *systemmessage*, for which additional explanation is available in the `intro`(2) man page.

`-` *xmit*`/`*total* `frames transmitted, total` *count* `lost`
`<atmtest>`

This message does not necessarily indicate any error. This informational message is printed once for every 1000 transmitted packets. The *xmit* value indicates the number of packets transmitted, *total* indicates the number of packets `atmtest` has been asked to transmit and receive, and *count* indicates the number of packets that have been lost (transmitted but not received), so far.

`ifatmconfig: bad ATM address.`
`<ifatmconfig>`

When `ifatmconfig` was attempting to parse the ATM address for the LIS's ATMARP server, it discovered that the address is not valid. Follow the description of valid entries provided in Table 16, page 114, before trying again.

`ifatmconfig: couldn't bind raw socket:` *systemmessage*
`<ifatmconfig>`

While processing a command line entry, the `ifatmconfig` command was unable to bind to a raw socket. This indicates a problem with the operating system (for example, it is overloaded). Nothing is wrong with the IRIS ATM software. The *systemmessage* (a standard system errno) for the failure indicates what the problem is. Additional explanation is available in the `intro`(2) man page.

`ifatmconfig: couldn't open a raw socket:` *systemmessage*
`<ifatmconfig>`

While processing a command line entry, the `ifatmconfig` command was unable to open a raw socket. This indicates a problem with the operating system (for example, it is overloaded). Nothing is wrong with the IRIS ATM software. The *systemmessage* (a standard system errno) for the failure indicates what the problem is. Additional explanation is available in the `intro`(2) man page.

`ifatmconfig -F : couldn't open input file:` *filename systemmessage*
`<ifatmconfig>`

The `ifatmconfig` command could not open the specified *filename* that it believes is the LIS configuration file. Before trying again, verify that the file exists. The default name for this file is `/var/atm/ifatm.conf`. The *systemmessage* (a standard system errno) for the failure indicates what the problem is. Additional explanation is available in the `intro`(2) man page.

`ifatmconfig -F : line` *linenumber* `file` *filename*
`couldn't bind raw socket:` *systemmessage*
`<ifatmconfig>`

While processing the specified entry (*linenumber*) in the LIS configuration file (*filename*), the `ifatmconfig` command was unable to bind to a raw socket. This indicates a problem with the operating system (for example, it is overloaded). Nothing is wrong with the IRIS ATM software. The *systemmessage* (a standard system errno) for the failure indicates what the problem is. Additional explanation is available in the `intro`(2) man page.

`ifatmconfig -F : line` *linenumber* `file` *filename*
`couldn't open a raw socket:` *systemmessage*
`<ifatmconfig>`

While processing the specified entry (*linenumber*) in the LIS configuration file (*filename*), the `ifatmconfig` command was unable to open a raw socket. This indicates a problem with the operating system (for example, it is overloaded). Nothing is wrong with the IRIS ATM software. The *systemmessage* (a standard system errno) for the failure indicates what the problem is. Additional explanation is available in the `intro`(2) man page.

`ifatmconfig -F : problem in line` *linenumber* `file` *filename*
`<ifatmconfig>`

While processing the LIS configuration file (*filename*), the `ifatmconfig` command was unable to process an entry on the specified line (*linenumber*). It is probable that a value provided after one of the key words is invalid. To resolve the problem, edit the file as explained in Section 3.5.5.2, page 111.

`ifatmconfig -F : syntax error line` *linenumber* `file` *filename*
`<ifatmconfig>`

While reading the LIS configuration file (`filename`), the `ifatmconfig` command encountered an illegal entry at the start of the specified line (*linenumber*). The first entry on each line must be the name of an IRIS ATM logical network interface (for example, `atm0` or `atm1`). To correct the problem, edit the file as explained in Section 3.5.5.2, page 111.

`ifatmconfig -F : syntax error line` *linenumber* `file` *filename*
`<ifatmconfig>`

While processing the LIS configuration file (*filename*), the `ifatmconfig` command encountered an illegal entry on the specified line (*linenumber*). To resolve the problem, edit the file as explained in Section 3.5.5.2, page 111.

`ifatmconfig: problem with` *userentry* `parameter`
`<ifatmconfig>`

The indicated command line entry (*userentry*) is not a valid entry.

`invalid addr %08`*hexnumeral* `in section` *number* `<`*filename*`>`
`<atmconfig>`

While attempting to download new firmware, `atmconfig -l` found that the indicated section (*number*) of the file containing the firmware (*filename*) has an invalid address. To resolve this, reinstall the IRIS ATM software and restart the driver.

`Listen ioctl failed:` *systemmessage*
`<sigtest>`

The driver was unable to associate itself with ("listen on") an incoming VC, as requested by `sigtest`. The `ioctl()` command `ATMIOC_LISTEN` failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Bad address	EFAULT: An error occurred when the ATM software was accessing the call's argument.
Interrupted system call	EINTR: While waiting for a request to appear on the queue, the call was interrupted unexpectedly.

No such device	ENODEV: The board was not in the UP or DOWN state. Or, the port was not operational.

`Location of failure =` *location_number* `(`*location_message*`)`
`<sigtest>`

The setup call was rejected at the location specified in the error message. ATM rejection locations are summarized in Table 31, page 207.

```
may take up to 1 minute for flash-burn to complete.
Finished.
<atmconfig>
```

This is not an error message. The message is only meant to warn you not to interrupt this process until the "Finished" message appears, since an interruption could cause corruption of the data on the board's EEPROM.

`MPSetup ioctl failed:` *systemmessage*
`<sigtest>`

The driver was unable to set up the multipoint VC, as requested by `sigtest`. The `ioctl()` command `ATMIOC_MPSETUP` failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Bad address	EFAULT: An error occurred when the ATM software attempted to read the call's argument.
Interrupted system call	EINTR: The driver was interrupted. The setup request cannot be completed. Try again.
Invalid argument	EINVAL: The file descriptor was already bound. or the access mode (read/write) was incorrect.
I/O error	EIO: The setup call was rejected by the network (an intermediate system) or by the called party. The reason (cause) for the rejection is provided in another error message.
No space left on device	ENOSPC: The driver was unable to allocate a unique identifier to the SVC. This may indicate that there are too many VCs open already.
No such device	ENODEV: The board was not in the UP or DOWN state.

Or, the port was not operational.

`No ARP server assigned`
`<ifatmconfig>`
While attempting to display information about an LIS, the `ifatmconfig` command retrieved a null address for the ATMARP server. This probably indicates that the logical network interface specified on the command line has not been configured (in the `/var/atm/ifatm.conf` file) with an ATMARP server.

`No ATM Address`
`<ifatmconfig>`
While attempting to display information about an LIS, the `ifatmconfig` command discovered that the ATM address is null. This means that the LIS does not have an assigned ATM address. If the adjacent switch supports address registration, this message indicates that there is a problem with the connection to the switch or with the switch itself. If the switch does not support address registration, configure this address by as explained in Section 3.7.1, page 126.

`no ATM interfaces available`
`<atmarp>`
There are no IRIS ATM network interfaces available (that is, configured and enabled). Since the `atmarp` utility binds lower-layer services (virtual circuits) to the upper-layers (that is, the IP protocol stack), it requires an operational IRIS ATM network interface.

Use `netstat -in` to display the current configuration of network interfaces. If no IRIS ATM network interface (`atm0`, `atm1`, etc.) is configured and enabled, follow the instructions in Section 3.5, page 104, to remedy this condition. If no IRIS ATM network interface is listed, use `versions` to verify that the IRIS ATM software is installed, use `autoconfig` to build the driver into the operating system, then reboot to start using the new operating system.

`Open failed:` *`systemmessage`*
`<sigtest>`
While attempting to create a write-only VC for transmission, the `open()` call for a file descriptor to the IRIS ATM subsystem failed. This probably indicates a configuration or hardware availability problem. The reason for the failure is provided in the *systemmessage*, for which additional explanation is available in the `intro`(2) man page. There is nothing wrong with IRIS ATM.

`Open for accept FD failed:` *systemmessage*
`<sigtest>`

The driver's attempt to start accepting data on an open, activated VC failed. That is, it could not open a new file descriptor to the device. The reason for the failure is provided in the *systemmessage*, for which additional explanation is available in the `intro`(2) man page.

`PANIC: atmintr: u=`*unit*`: illegal b2h cmd: 0x`*hexnumber* `@ 0x`*address*
`<driver>`

When processing an interrupt from the board, the driver discovered a serious problem with its communication mechanism to the board. The command from the board (0x*hexnumber*) is not a recognized one. This may indicate a mismatch between the driver and board firmware versions. Follow the instructions in Section 3.9, page 131, to reset the indicated board (*unit*) and then restart the driver. Restarting the driver with `atmconfig` causes the versions to be compared and a firmware update to be done if necessary.

`PANIC: atmintr: u=`*unit*`: xcmd->cmd != b2h->cq_cmd! xcmd=0x`*hexnumber* `b2h=0x`*hexnumber*
`<driver>`

When processing an interrupt from the board, the driver discovered a serious problem with its communication mechanism to the board. The driver expected the commands on the two communication queues to match (xcmd = b2h), however they did not. The actual commands encountered are displayed as hexadecimal numbers. Follow the instructions in Section 3.9, page 131, to reset the indicated board (*unit*). If the problem persists, contact the Silicon Graphics Technical Assistance Center or the site's support engineer.

`PANIC: couldn't allocate stats area, unit=`*number*
`<driver>`

During startup, the driver was unable to allocate memory for use in maintaining status information for the indicated board (*number*). This may indicate a problem with system memory. This does not indicate any problem with the IRIS ATM board or software.

`PANIC: couldn't create if_atm interface` *interfacenumber*
`<driver>`

The driver was unable to allocate memory for the IRIS ATM interface indicated (*number*). This indicates a problem with the operating system.

`PANIC: couldn't kmem_zalloc atm unit` *number*
`<driver>`

During startup, the driver was unable to allocate memory for use in managing information about the indicated board (*number*). This may indicate a problem with system memory. This does not indicate any problem with the IRIS ATM board or software.

`PANIC: couldn't kvpalloc b2h, unit=`*number*
`<driver>`

During startup, the driver was unable to allocate memory for use in managing the communication interface to the indicated board (*number*). This may indicate a problem with system memory. This does not indicate any problem with the IRIS ATM board or software.

`PANIC: couldn't kvpalloc cmdq, unit=`*number*
`<driver>`

During startup, the driver was unable to allocate memory for use in managing the communication interface to the indicated board (*number*). This may indicate a problem with system memory. This does not indicate any problem with the IRIS ATM board or software.

`PANIC: couldn't kvpalloc hca area, unit=`*number*
`<driver>`

During startup, the driver was unable to allocate memory for use in managing the communication interface to the indicated board (*number*). This may indicate a problem with system memory. This does not indicate any problem with the IRIS ATM board or software.

`PANIC: unknown input buffer type` *number*
`<driver>`

When attempting to read a buffer containing incoming (received) data, the driver encountered an unknown buffer size (or type). The displayed *number* indicates the type of buffer encountered. This may indicate a mismatch between the versions of board firmware and operating system driver.

To remedy this, reinstall the IRIS ATM software, rebuild the operating system to include the new driver, and reboot the machine. The new driver may download new firmware onto the board during the reboot. If the problem persists, contact the Silicon Graphics Technical Assistance Center or the site's support engineer.

`port  (unbound)`
`<ifatmconfig>`

While attempting to display information about an LIS, the `ifatmconfig` command could not retrieve the port assigned to the LIS. This may indicate that

the logical network interface specified on the command line has not been configured (in the `/var/atm/ifatm.conf` file) with a port.

`progmac: load_firm(): trouble with # ATMIOC_STO:` *systemmessage*`),`
`<atmconfig>`

When attempting to download firmware onto the board's EEPROM, `atmconfig` was unable to complete the task. The `ioctl()` command `ATMIOC_STO` failed. The pound sign (#) indicates whether the error occurred on the first or second instance of this call. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Bad address	EFAULT: An error occurred during a data copy. Try again.
No permission match	EPERM: The process must have super-user access privileges.
Timer expired	ETIME: The board is not responding in a timely manner. It may be asleep or dysfunctional.

`read failed for section` *number* *<filename>*
`<atmconfig>`

While attempting to download new firmware, `atmconfig -l` could not read the indicated section (*number*) in the file containing the firmware. To resolve this, reinstall the IRIS ATM software.

`Registration ioctl failed:` *systemmessage*
`<sigtest>`

The driver was unable to register to receive incoming VC requests, as requested by `sigtest`. The `ioctl()` command `ATMIOC_REGISTER` failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Bad address	EFAULT: An error occurred when the ATM software attempted to read the call's argument.
Invalid argument	EINVAL: The file descriptor was already bound, the access mode (read/write) was incorrect, or an internal value was invalid. This indicates a problem with the IRIS ATM software. Follow the instructions to gracefully bring the software down, then restart it.

I/O error	EIO: The registration request was rejected at the other endpoint or along the ATM network, and the reason (cause) has been provided in another error message.
No space left on device	ENOSPC: The driver was not able to allocate an internal identifier to the SVC. This may indicate that too many VCs are already open.
No such device	ENODEV: The board was not in the UP or DOWN state. Or, the port was not operational.

`S2D_RELEASE_IND: unused state. crossed releases? userHandle=0x`*hexnumber*
`<signaling software>`

An attempt to release an SVC failed. The hexadecimal number displayed indicates the local (software internal) identification number for that VC.

`section` *number* `not in file <`*filename*`>`
`<atmconfig>`

While attempting to download new firmware, `atmconfig -l` could not locate the indicated section (*number*) in the file containing the firmware. To resolve this, reinstall the IRIS ATM software.

`section` *number* `<`*filename*`> too big,` *actualsize* `>=` *maxsize*
`<atmconfig>`

While attempting to download new firmware, `atmconfig -l` found that the indicated section (*number*) of the file containing the firmware (*filename*) was larger than allowed (*maxsize*). To resolve this, reinstall the IRIS ATM software.

`Setup ioctl failed:` *systemmessage*
`<sigtest>`

The driver was unable to set up the VC, as requested by `sigtest`. The `ioctl()` command `ATMIOC_SETUP` failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Bad address	EFAULT: An error occurred when the ATM software attempted to read the call's argument.
Interrupted system call	EINTR: The driver was interrupted and the setup request cannot be completed. Try again.
Invalid argument	EINVAL: The file descriptor was already bound. Or the access mode (read/write) was incorrect.

I/O error	EIO: The setup call was rejected by the network (an intermediate system) or by the called party. The reason (cause) for the rejection is explained in another error message.
No space left on device	ENOSPC: The driver was unable to allocate a unique identifier to the SVC. This may indicate that there are too many VCs open already.
No such device	ENODEV: The board was not in the UP or DOWN state. Or, the port was not operational.

`Setup : memalign obuf failed:` *systemmessage*`.`
`<sigtest>`

While preparing to send data, `sigtest` was unable to allocate (`memalign`) memory for its outgoing data. This indicates a problem with the operating system. For example, all memory is occupied by other processes. The reason for the failure is provided in the *systemmessage*, for which additional explanation is available in the `intro`(2) man page. There is nothing wrong with IRIS ATM.

`Setup : mpin of obuf failed:` *systemmessage*`.`
`<sigtest>`

While preparing to send data, `sigtest` was unable to lock down (`mpin`) enough memory to handle its outgoing data. This probably indicates that *sigtest* has exceeded its allowable limit for locked memory. The reason for the failure is provided in the *systemmessage*, for which additional explanation is available in the `intro`(2) man page. There is nothing wrong with IRIS ATM.

`This interface has no MAC address.`
`<atmconfig>`

While attempting to read the MAC address from the board, `atmconfig -m` discovered the board does not have one. IRIS ATM cannot operate without a MAC address. Contact the Silicon Graphics Technical Assistance Center or the site's support engineer.

`too many sections in <`*filename*`>` *number* `>=` *max_sections_allowed*
`<atmconfig>`

While attempting to download new firmware, `atmconfig -l` found that the indicated file has the indicated number of sections (*number*), which exceeds the maximum number allowed. To resolve this, reinstall the IRIS ATM software.

`trouble with ATMIOC_CONTROL down:` *systemmessage*
`<atmconfig>`

While trying to initialize the board into the DOWN state, `atmconfig -d` encountered an error. The `ioctl()` command `ATMIOC_CONTROL` for initializing the board failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page. For more information on the use of the `-d` option, see Section 3.2.2.2, page 86, for Origin2000 or Onyx 2 systems, or Section 3.3.4.3, page 100, for Challenge or Onyx systems.

Device busy	EBUSY: The board is not responding. It may be dysfunctional or frozen. Invoke `atmconfig -r` followed by `atmconfig -d` to manually put the board into the DOWN state. If this does not work, power cycle the machine. For more information on the use of these options, see Section 3.2.2.2, page 86, for Origin2000 or Onyx 2 systems, or Section 3.3.4.3, page 100, for Challenge or Onyx systems.
Invalid argument	EINVAL: The board is not in the appropriate state. For example, in order to bring the board to the DOWN state, the board must first be in the PRE-INIT state, and in order to bring the board UP, the board must first be in the DOWN state.
I/O error	EIO: The initialization routine failed to bring the board to the DOWN state.
No permissions match	EPERM: You must be super user (root).

`trouble with ATMIOC_CONTROL init:` *systemmessage*
`<atmconfig>`

After downloading new firmware and resetting the board, `atmconfig -d` could not initialize the board. The `ioctl()` command `ATMIOC_CONTROL` for initialization failed.The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.For more information about the use of the `-d` option, see Section 3.2.2.1, page 86.

Device busy	EBUSY: The board is not responding. It may be dysfunctional or frozen. Invoke `atmconfig -r` followed by `atmconfig -d` to manually put the board into the DOWN state. If this does not work, power cycle the machine. For more information on the use of these options, see

	Section 3.2.2.2, page 86, for Origin2000 or Onyx 2 systems, or Section 3.3.4.3, page 100, for Challenge or Onyx systems.
I/O error	EIO: The initialization routine failed to bring the board to the DOWN state.
Invalid argument	EINVAL: The board is not in the appropriate state. For example, in order to bring the board to the DOWN state, the board must first be in the PRE-INIT state, and in order to bring the board UP, the board must first be in the DOWN state.
No permissions match	EPERM: You must be super user (root).

`trouble with ATMIOC_CONTROL reset:` *systemmessage*
`<atmconfig>`

After downloading new firmware, `atmconfig -d` attempted to reset the board and encountered a problem. The `ioctl()` command `ATMIOC_CONTROL` for reset failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page. For more information about the use of the `-d` option, see Section 3.2.2.1, page 86.

No permissions match	EPERM: You must be super user (root).

`trouble with ATMIOC_CONTROL reset:` *systemmessage*
`<atmconfig>`

While attempting to reset the board, `atmconfig -r` encountered a problem. The `ioctl()` command `ATMIOC_CONTROL` for reset failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

No permissions match	EPERM: You must be super user (root).

`trouble with ATMIOC_CONTROL reset (down):` *systemmessage*
`<atmconfig>`

While attempting to bring the board to the DOWN state, `atmconfig -d` found that the board was not in the pre-initialized state, so it attempted to reset the board. The `ioctl()` command `ATMIOC_CONTROL` for reset failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page. For more information about the use of the `-d` option, see Section 3.2.2.1, page 86.

No permissions match	EPERM: You must be super user (root).

`trouble with ATMIOC_CONTROL up:` *systemmessage*
`<atmconfig>`

When trying to put the board into the UP state, `ifconfig -u` encountered a problem. The `ioctl()` command `ATMIOC_CONTROL` for up failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Invalid argument	EINVAL: The board is not in the appropriate state. For example, in order to bring the board to the DOWN state, the board must first be in the PRE-INIT state, and in order to bring the board UP, the board must first be in the DOWN state.
No permissions match	EPERM: You must be super user (root).
Timer expired	ETIME: The board is not responding in a timely manner. It may be asleep or dysfunctional.

`trouble with ATMIOC_GETMACADDR:` *systemmessage*
`<atmconfig>`

While attempting to read the MAC address from the board, `atmconfig -m` encountered a problem. The `ioctl()` command `ATMIOC_GETMACADDR` failed. The *systemmessage* (a standard system errno) for the failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Bad address	EFAULT: An error occurred during a data copy. Try the command again.
No such device	ENODEV: One of the boards mentioned in the IP-to-ATM mapping file is not in the UP state.
Timer expired	ETIME: The board did not process the command and a relatively long period of time passed. The board may be very busy or may not be functional.

`trouble with ATMIOC_VERS:` *systemmessage*
`<atmconfig>`

While attempting to retrieve the firmware version from the board, `atmconfig -V` encountered a problem. The `ioctl()` command `ATMIOC_VERS` failed. The *systemmessage* (a standard system errno) for the

failure indicates what the problem is, as summarized in the following list. Additional explanation is available in the `intro`(2) man page.

Bad address	EFAULT: An error occurred during a data copy. Try the command again.
Timer expired	ETIME: The board did not process the command and a relatively long period of time passed. The board may be very busy or may not be functional.

`Unknown Address Type` *userentry*
`<sigtest>`
The *userentry* is invalid. Use only digits displayed on the menu.

`Unknown type` *userentry*
`<sigtest>`
The *userentry* that was used for selecting a type of cellrate is not valid. Use only the digits displayed in the menu.

`Unparsable:` *userentry*
`<atmconfig>`
While attempting to write a MAC address into the board's FLASH EPROM, the command could not parse the indicated *userentry* that was supplied on the command line. The user entry must consist of 12 hexadecimal characters (0-9, a-f, or A-F) with or without separating colons and no blank spaces between the characters. For example, 09C8715AF983 and 09:c8:71:5a:f9:83 are both valid.

`''`*userentry*`'' is not a number, try again:`
`<sigtest>`
The user supplied an invalid *userentry*. The entry must be a digit (number): 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9.

`''`*userentry*`'' is not a valid choice.`
`<sigtest>`
The specified *userentry* is not an acceptable (valid) entry for the line. Use only the values described in the menu.

message `/var/atm/atmsigd.tcl` *linenumber*
`(signaling software)`
While attempting to read its configuration file, the signaling software was unable to parse an entry located on the indicated line (*linenumber*). The *message* describes the nature of the problem. Due to this problem, the signaling software was not able to configure itself and start. To resolve this problem, edit the

/var/tmp/atmsigd.tcl file as described in Section 3.6.1, page 123, then restart the IRIS ATM signaling software as described in Table 19, page 132.

```
WARNING: atmarp: AAOP_ARP_REQUEST: ARP table full!
<driver>
```

While attempting to add or update an entry in the local ATMARP table, the driver encountered an error due to the table being already full. If the system is configured to perform as the ATMARP server for one or more LISs, it is possible that there are too many hosts. In this case, select a different system to be ATMARP server for a number of the LISs in order to remove the overload from this system. If the system is not an ATMARP server, ifconfig one or more logical network interfaces down or remove some entries from the IP-over-PVC address configuration file (/var/atm/pvc.conf), in order to free space in the table.

```
WARNING: atmarp: AAOP_InARP_REPLY: ARP table full!
<driver>
```

While attempting to add an entry (IP address) for an ATMARP server to the local ATMARP table, the driver encountered an error due to the table being already full. If the system is configured to perform as the ATMARP server for one or more LISs, it is possible that there are too many hosts. In this case, select a different system to be the ATMARP server for a number of the LISs in order to remove the overload from this system. If the system is not an ATMARP server, ifconfig one or more logical network interfaces down or remove some entries from the IP-over-PVC address configuration file (/var/atm/pvc.conf), in order to free space in the table.

```
WARNING: atmarp_input: atm#: received ATMARP_REQUEST but not server.
<driver>
```

The IRIS ATM software received an ATMARP request for resolution of an IP-to-ATM address, however, this system is not configured as the ATMARP server for the LIS indicated (atm#). This probably indicates that some endpoint has been configured to use this station as its server. There is nothing wrong with the IRIS ATM subsystem.

```
WARNING: atmarp_input: atm#: received ATM InARP_REPLY but not server.
<driver>
```

The IRIS ATM software received an Inverse ATMARP request for resolution of an ATM-to-IP address, however, this system is not configured as the ATMARP server for the LIS indicated (atm#). This probably indicates that some endpoint has been configured to use this station as its server. There is nothing wrong with the IRIS ATM subsystem.

```
WARNING: atmarp: REPLY: ARP table full!
<driver>
```

While attempting to add the results from an address resolution request to the local ATMARP table, the driver encountered an error due to the table being already full. If the system is configured to perform as the ATMARP server for one or more LISs, it is possible that there are too many hosts. In this case, select a different system to be ATMARP server for a number of the LISs in order to remove the overload from this system. If the system is not an ATMARP server, `ifconfig` one or more logical network interfaces down or remove some entries from the IP-over-PVC address configuration file (`/var/atm/pvc.conf`), in order to free space in the table.

```
WARNING: atmclose: couldn't destroy fwd VCTE, error = number
<driver>
```

While closing a transmit virtual channel, a problem occurred at the board level, and the VC could not be dismantled. The error (*number*) supplied by the board indicates the nature of the problem. This message may indicate that an upper-layer control program asked the driver to tear down a VC that did not exist or that was only open for receiving (not for sending).

```
WARNING: atmclose: couldn't destroy rvs VCTE
<driver>
```

While closing a receive virtual channel, a problem occurred at the board level, and the VC could not be dismantled. This message may indicate that an upper-layer control program asked the driver to tear down a VC that did not exist or that was only open for transmitting (not for receiving).

```
WARNING: atminit: duplicate unit # in slot number adap number will be
ignored.
first unit # is in slot number adap number.
<driver>
```

During power up, two IRIS ATM boards were encountered that have the same unit number. This error can only occur when the following two conditions are true:

- The `/var/sysgen/master.d/atm` file is configured to read the unit jumper set on each board.
- The unit jumper set on two installed boards are both configured with the same settings.

To remedy this problem do either one of the following: (1) edit the `/var/sysgen/master.d/atm` file to ignore the settings on the jumpers and

reboot, or (2) remove one of the boards indicated in the error message and reconfigure its unit jumper setto a unique setting, as described in the hardware's installation instructions.

`WARNING: atmintr (`*unitnumber*`): couldn't clear INT1`
`<driver>`

While processing an interrupt from the board, the driver discovered it could not clear the interrupt. This may indicate a problem with the board's GIO Interrupt Register. Use `atmconfig` to reset the board. If the problem persists, contact the Silicon Graphics Technical Assistance Center or the site's support engineer.

`WARNING: atm_odone: couldn't destroy fwd VCTE, error =` *number*
`<driver>`

The board transmitted the final frame for a virtual circuit that the upper-layer control program has asked the driver to tear down. The driver's request for the board to free up the resources for that VC failed. The error (*number*) indicates the reason.

`WARNING: atm_wait_hca: TIMED OUT! unit=`*number*
`<driver>`

While waiting for the indicated board (*number*) to process a driver-to-board command, the driver waited long enough to believe that something is wrong with the board. The board did not process the command and a relatively long period of time passed. In many instances, subsequent error messages indicate more about the problem. The board may have encountered a problem that caused it to freeze or it may be dysfunctional.

Use the `atmconfig` command to transition the board from its current state to DOWN, and then to UP. If the problem persists, use `atmconfig` to reset the board. If this message is displayed again, shut down the system and turn off the power; then, power it on. If this message is displayed again, contact the Silicon Graphics Technical Assistance Center or the site's support engineer.

`WARNING: atm_xmit_dn(`*unitnumber*`): XCMD_XMIT result =` *errornumber*
`<driver>`

An error occurred during a transmission on the indicated board (*unitnumber*). The error (*errornumber*) provides further details.

`WARNING: couldn't bring up board, unit=`*number*
`<driver>`

While attempting to bring the indicated board (*number*) to the initialized (DOWN) state, the driver was not able to initialize it. The board may have encountered a problem that caused it to freeze or it may be dysfunctional.

Invoke `atmconfig -r` to reset the board, then use `atmconfig -d` to manually put the board into the DOWN state. If this does not work, shut down the system and turn off the power; then, power it on. If this message is displayed again, contact the Silicon Graphics Technical Assistance Center or the site's support engineer. For more information on the use of these options, see Section 3.2.2.2, page 86, for Origin2000 or Onyx 2 systems, or Section 3.3.4.3, page 100, for Challenge or Onyx systems.

`WARNING: couldn't post LARGE bufs, unit=`*unitnumber* `error=`*errornumber*
`<driver>`

While passing new large-sized buffers to the board, a problem occurred and the buffers were not accepted by the board. The error supplied by the board (*errornumber*) indicates the problem.

`WARNING: couldn't post MED bufs, unit=`*number* `error=`*errornumber*
`<driver>`

While passing new medium-sized buffers to the board, a problem occurred and the buffers were not accepted by the board. The error supplied by the board (*errornumber*) indicates the problem.

`WARNING: couldn't post SML bufs, unit=`*unitnumber* `error=`*errornumber*
`<driver>`

While passing new small-sized buffers to the board, a problem occurred and the buffers were not accepted by the indicated board (*digit*). The error supplied by the board (*errornumber*) indicates the problem.

`WARNING: dang_intr_conn() failed for unit` *number*
`<driver>`

During startup, the driver failed to contact the DANG device on the ATM-OC3c board. This indicates a hardware problem. Contact the Silicon Graphics Technical Assistance Center or the site's support engineer.

`WARNING: S2D_REGRESPONSE: register response failed for IP. status =` *status*
`<signaling>`

While the driver was attempting to place a message on the IRIS ATM signaling software's queue, an error occurred. The affected packet is an IP packet. The specific message is `S2D_REGRESPONSE`. When *status* is a nonzero value, the

message has failed to complete successfully. This may indicate a problem with `atmsigd`.

```
WARNING! Writing ATM interface FLASH_PROM.
Do not reset the system until finished
may take up to 1 minute for flash-burn to complete.
Finished.<atmconfig>
```

This is not an error message. The message is only meant to warn you not to interrupt this process since an interruption could cause corruption of the data on the board's EEPROM.

`Write failed:` *systemmessage*
`<sigtest>`

When `sigtest` wrote the outgoing data to its file descriptor, the call failed. The reason for the failure is provided in the *systemmessage*, for which additional explanation is available in the `intro`(2) man page. There is nothing wrong with IRIS ATM.

Causes and Diagnostics [A]

This appendix describes the results that are returned with ATM signaling requests. The cause codes and diagnostic information that are described are returned when an ATM request is rejected at the other endpoint or anywhere along the connection. The numerical values for ATM UNI causes and diagnostics are those assigned by the ATM standards; the values for Silicon Graphics messages are specific to IRIS ATM.

Table 29 lists the cause codes that are used by implementations that conform to the *ATM User-Network Interface Specification* (ATM UNI) standard. The Comments column points out codes that are specific to particular versions of the ATM UNI (for example, 3.0 and 3.1). Table 32, page 208, summarizes the diagnostic information that accompany some of the cause codes. Table 30, page 206, lists implementation-specific cause codes used by the IRIS ATM Signaling software.

Table 29. ATM UNI Cause Codes

Text for ATM UNI Cause	Cause Number	Comments
`Unallocated / Unassigned Number`	1	Additional information is supplied. See Table 32, page 208.
`No Route to Specified Transit Network`	2	
`No Route to Destination`	3	Additional information is supplied. See Table 32, page 208.
`Unacceptable VPCI_VCI`	10	
`Normal_3.1`	16	Not used with UNI 3.0 Used only with UNI 3.1
`User Busy`	17	
`No User Responding`	18	

Text for ATM UNI Cause	Cause Number	Comments
Call Rejected	21	Additional information is supplied. See Table 32, page 208.
Number Changed	22	Additional information is supplied. See Table 32, page 208.
User Rejects Calls With Calling Line Identification Restriction (CLIR)	23	
Destination Out of Order	27	
Invalid Number Format	28	
Response to STATUS ENQUIRY	30	
Normal_3.0	31	Used only with UNI 3.0 Not used with UNI 3.1
Requested VPCI/VCI Unavailable	35	
VPCI Assignment Failure	36	
User Cell Rate Unavailable	37	Not used with UNI 3.0 Used only with UNI 3.1
Network Out of Order	38	
Temporary Failure	41	
Access Information Discarded	43	Additional information is supplied. See Table 32, page 208.
No VPCI/VCI Available	45	
Resource Unavailable, Unspecified	47	
QOS Unavailable	49	Additional information is supplied. See Table 32, page 208.

Text for ATM UNI Cause	Cause Number	Comments
User Cell Rate Unavailable	51	Used only with UNI 3.0 Not used with UNI 3.1 Additional information is supplied. See Table 32, page 208.
Bearer Capability Not Authorized	57	
Bearer Capability Not Presently Available	58	
Service or Option Unavailable, Unspecified	63	
Bearer Capability Not Implemented	65	
Unsupported Combination of Traffic Parameters	73	
AAL Parameters Cannot Be Supported	78	Not used with UNI 3.0 Used only with UNI 3.1
Invalid Call Reference	81	
Identified Channel Does Not Exist	82	Additional information is supplied. See Table 32, page 208.
Incompatible Destination	88	Additional information is supplied. See Table 32, page 208.
Invalid Endpoint Reference	89	
Invalid Transit Network Selection	91	
Too Many Pending Add Party Requests	92	
AAL Parameters Cannot Be Supported	93	Used only with UNI 3.0 Not used with UNI 3.1

Text for ATM UNI Cause	Cause Number	Comments
Mandatory Information Element Missing	96	Additional information is supplied. See Table 32, page 208.
Message Type Non-existent or Not Implemented	97	Additional information is supplied. See Table 32, page 208.
Information Element Non-existent or Not Implemented	99	Additional information is supplied. See Table 32, page 208.
Invalid Information Element Contents	100	Additional information is supplied. See Table 32, page 208.
Message Not Compatible With Call State	101	Additional information is supplied. See Table 32, page 208.
Recovery On Timer Expiry	102	Additional information is supplied. See Table 32, page 208.
Incorrect Message Length	104	
Protocol Error, Unspecified	111	

Table 30. SGI Cause Codes

Text for SGI Cause	Cause Number	Comments
Local Error	128	Unknown driver or signal-daemon error
Already	129	Registration denied: BLLI already taken, or application already registered
Invalid Best Effort	130	Best Effort requires that both directions be Best Effort and QOS_0

Text for SGI Cause	Cause Number	Comments
`Invalid Cell Rate`	131	Invalid `cellrate` field
`Invalid BLLI`	132	Invalid broadband low layer information (`blli`) code specified
`Invalid Bearer Class`	133	Invalid bearer class
`Invalid Dress FMT`	134	Invalid address format
`Not Multipoint`	135	Add or drop party on a point-to-point call
`Party Handle In Use`	136	Trying to add a party using the a party handle that has already been used
`Invalid Party Handle`	137	Request was dropped because the party handle was not found

Table 31. ATM UNI Failure Locations

Text for ATM UNI Location	Location Number	Comments
`User`	0x00	Local host
`Private Network Serving the Local User`	0x01	Private local switch
`Public Network Serving the Local User`	0x02	Public local switch
`Transit Network`	0x03	Transit network
`Public Network Serving the Remote User`	0x04	Remote public switch

Text for ATM UNI Location	Location Number	Comments
`Private Network Serving the Remote User`	0x05	Remote private switch
`International Network`	0x07	International network
`Network Beyond Interworking Point`	0x0A	Network beyond interworking point

Table 32. ATM UNI Diagnostics

Accompanying ATM UNI Cause	ATM UNI Diagnostic Provided	Diagnostic Values
Unallocated / Unassigned Number	One octet	The diagnostics provide a number to indicate the condition, and text to indicate whether the condition is normal or abnormal and who supplied the diagnostic, as follows: 0x80 Unknown normal provider 0x81 Permanent normal provider 0x82 Transient normal provider 0x84 Unknown abnormal provider 0x85 Permanent abnormal provider 0x86 Transient abnormal provider 0x88 Unknown normal user 0x89 Permanent normal user 0x8A Transient normal user 0x8C Unknown abnormal user 0x8D Permanent abnormal user 0x8E Transient abnormal user
`Call Rejected`	Two octets	The diagnostics provide the following information: the first octet contains a number indicating the reason, and a description of the condition. The second octet contains either user-specific values or the identifier for the ATM UNI information element (IE), whichever is appropriate.

Accompanying ATM UNI Cause	ATM UNI Diagnostic Provided	Diagnostic Values
		0x80 user specific unknown 0x81 user specific permanent 0x82 user specific transient 0x84 IE missing unknown 0x85 IE missing permanent 0x86 IE missing transient 0x88 IE missing unknown 0x89 IE missing permanent 0x8A IE missing transient
`No Route to Destination`	One octet	Same as `Unallocated Number`.
`Number Changed`	6 to 25 octets	The new destination address formatted with a Called Party Number information element.
`Access Information Discarded`	One or more octets	Each octet specifies one ATM UNI information element identifier.
`QOS Unavailable`	One octet	Same as `Unallocated Number`.
`User Cell Rate Unavailable`	One or more octets	Each octet specifies one subfield identifier from the ATM UNI User Cell Rate information element.
`Identified Channel Does Not Exist`	4 octets	Most significant two octets specify VPCI value. Least significant two octets specify VCI value.
`Incompatible Destination`	1 octet	The ATM UNI information element identifier.
`Mandatory Information Element Missing`	1 or more octets	Each octet is one ATM UNI information element identifier.
`Message Type Non-existent or Not Implemented`	One octet	Specifies one ATM UNI message type: for example, SETUP, RELEASE, CONNECT.
`Information Element Non-existent or Not Implemented`	1 or more octets	Each octet is one ATM UNI information element identifier.
`Invalid Information Element Contents`	1 or more octets	Each octet is one ATM UNI information element identifier.

Accompanying ATM UNI Cause	ATM UNI Diagnostic Provided	Diagnostic Values
`Message Not Compatible With Call State`	One octet	Specifies one ATM UNI message type: for example, SETUP, RELEASE, CONNECT.
`Recovery On Timer Expiry`	Three octets	Each octet specifies one IA5 character to indicate one numeral identifying an ATM UNI timer. For example, for the timer called `T308`, the first octet specifies 3, the second 0, and the third 8.

Supported Transmission Rates for IRIS ATM Board on CHALLENGE and Onyx Platforms [B]

The tables in this appendix list the transmission rates (in megabits per second) supported by the ATM-OC3c for CHALLENGE and Onyx systems. Each rate queue on an IRIS ATM-OC3c HIO mezzanine board can be configured to any of the rates listed in this appendix.

Table 33. Supported Rates in Payload Bits Per Second: 137 to 13 Mbps

Megabits	(per second)				
137.1428	53.3333	32.0000	23.4146	18.4615	15.2381
128.0000	51.8919	31.4754	23.1325	18.2857	15.1181
120.0000	50.5263	30.9677	22.8571	18.1132	15.0000
112.9412	49.2308	30.4762	22.5882	17.9439	14.8837
106.6667	48.0000	30.0000	22.3256	17.7778	14.7692
101.0526	46.8293	29.5385	22.0690	17.6147	14.6565
96.0000	43.6364	29.0909	21.8182	17.4545	14.5455
91.4286	42.6667	28.6567	21.5730	17.2973	14.4361
87.2727	41.7391	28.2353	21.3333	17.1429	14.3284
83.4783	40.8511	27.8261	21.0989	16.9912	14.2222
80.0000	40.0000	27.4286	20.8696	16.8421	14.1176
76.8000	39.1837	27.0423	20.6452	16.6957	14.0146
73.8462	38.4000	26.6667	20.4255	16.5517	13.9130
71.1111	37.6471	26.3014	20.2105	16.4103	13.8129
68.5714	36.9231	25.9459	20.0000	16.2712	13.7143
66.2069	36.2264	25.6000	19.7938	16.1345	13.6170

Megabits	(per second)				
64.0000	35.5556	25.2632	19.5918	16.0000	13.5211
61.9355	34.9091	24.9351	19.3939	15.8678	13.4266
60.0000	34.2857	24.6154	19.2000	15.7377	13.3333
58.1818	33.6842	24.3038	19.0099	15.6098	13.2414
56.4706	33.1034	24.0000	18.8235	15.4839	13.1507
54.8571	32.5424	23.7037	18.6408	15.3600	13.0612

Table 34. Supported Rates in Payload Bits Per Second: 12.9 to 5 Mbps

Megabits	(per second)					
12.9730	11.2941	9.9482	8.9302	8.1013	7.2727	5.4545
12.8859	11.2281	9.8969	8.8889	8.0672	7.1642	5.3933
12.8000	11.1628	9.8462	8.8479	8.0335	7.0588	5.3333
12.7152	11.0983	9.7959	8.8073	8.0000	6.9565	5.2747
12.6316	11.0345	9.7462	8.7671	7.9668	6.8571	5.2174
12.5490	10.9091	9.6970	8.7273	7.9339	6.7606	5.1613
12.4675	10.8475	9.6482	8.6878	7.9012	6.6667	5.1064
12.3871	10.7865	9.6000	8.6486	7.8689	6.5753	5.0526
12.3077	10.7263	9.5522	8.6099	7.8367	6.4865	5.0000
12.2293	10.6667	9.5050	8.5714	7.8049	6.4000	
12.1519	10.6077	9.4581	8.5333	7.7733	6.3158	
12.0755	10.5495	9.4118	8.4956	7.7419	6.2338	
12.0000	10.4918	9.3659	8.4581	7.7108	6.1538	
11.9255	10.4348	9.3204	8.4211	7.6800	6.0759	
11.8519	10.3784	9.2754	8.3843	7.6494	6.0000	
11.7791	10.3226	9.2308	8.3478	7.6190	5.9259	
11.7073	10.2674	9.1866	8.3117	8.3117	5.8537	
11.6364	10.2128	9.1429	8.2759	7.5889	5.7831	

Megabits	(per second)				
11.5663	10.1587	9.0995	8.2403	7.5591	5.7143
11.4970	10.1053	9.0566	8.2051	7.5294	5.6471
11.4286	10.0524	9.0141	8.1702	7.5000	5.5814
11.3609	10.0000	8.9720	8.1356	7.3846	5.5172

Table 35. Supported Rates in Payload Bits Per Second: 4.9 to 2.1 Mbps

Megabits	(per second)				
4.9485	4.0000	3.3566	2.8916	2.5397	2.2642
4.8980	3.9669	3.3333	2.8743	2.5263	2.2535
4.8485	3.9344	3.3103	2.8571	2.5131	2.2430
4.8000	3.9024	3.2877	2.8402	2.5000	2.2326
4.7525	3.8710	3.2653	2.8235	2.4870	2.2222
4.7059	3.8400	3.2432	2.8070	2.4742	2.2120
4.6602	3.8095	3.2215	2.7907	2.4615	2.2018
4.6154	3.7795	3.2000	2.7746	2.4490	2.1918
4.5714	3.7500	3.1788	2.7586	2.4365	2.1818
4.5283	3.7209	3.1579	2.7429	2.4242	2.1719
4.4860	3.6923	3.1373	2.7273	2.4121	2.1622
4.4444	3.6641	3.1169	2.7119	2.4000	2.1525
4.4037	3.6364	3.0968	2.6966	2.3881	2.1429
4.3636	3.6090	3.0769	2.6816	2.3762	2.1333
4.3243	3.5821	3.0573	2.6667	2.3645	2.1239
4.2857	3.5556	3.0380	2.6519	2.3529	2.1145
4.2478	3.5294	3.0189	2.6374	2.3415	2.1053
4.2105	3.5036	3.0000	2.6230	2.3301	
4.1739	3.4783	2.9814	2.6087	2.3188	
4.1379	3.4532	2.9630	2.5946	2.3077	

Megabits	(per second)			
4.1026	3.4286	2.9448	2.5806	2.2967
4.0678	3.4043	2.9268	2.5668	2.2857
4.0336	3.3803	2.9091	2.5532	2.2749

Table 36. Supported Rates in Payload Bits Per Second: 2.0 to 0.7 Mbps

Megabits	(per second)				
2.0961	1.9048	1.4458	1.1321	0.9302	0.7895
2.0870	1.8972	1.4286	1.1215	0.9231	0.7843
2.0779	1.8898	1.4118	1.1111	0.9160	0.7792
2.0690	1.8824	1.3953	1.1009	0.9091	0.7742
2.0601	1.8750	1.3793	1.0909	0.9023	0.7692
2.0513	1.8462	1.3636	1.0811	0.8955	0.7643
2.0426	1.8182	1.3483	1.0714	0.8889	0.7595
2.0339	1.7910	1.3333	1.0619	0.8824	0.7547
2.0253	1.7647	1.3187	1.0526	0.8759	0.7500
2.0168	1.7391	1.3043	1.0435	0.8696	0.7453
2.0084	1.7143	1.2903	1.0345	0.8633	0.7407
2.0000	1.6901	1.2766	1.0256	0.8571	0.7362
1.9917	1.6667	1.2632	1.0169	0.8511	0.7317
1.9835	1.6438	1.2500	1.0084	0.8451	0.7273
1.9753	1.6216	1.2371	1.0000	0.8392	0.7229
1.9672	1.6000	1.2245	0.9917	0.8333	0.7186
1.9592	1.5789	1.2121	0.9836	0.8276	0.7143
1.9512	1.5584	1.2000	0.9756	0.8219	0.7101
1.9433	1.5385	1.1881	0.9677	0.8163	0.7059
1.9355	1.5190	1.1765	0.9600	0.8108	0.7018
1.9277	1.5000	1.1650	0.9524	0.8054	

Megabits	(per second)			
1.9200	1.4815	1.1538	0.9449	0.8000
1.9124	1.4634	1.1429	0.9375	0.7947

Table 37. Supported Rates in Payload Bits Per Second: 0.6 to 0.27 Mbps

Megabits	(per second)				
0.6977	0.6154	0.5505	0.4979	0.4348	0.3261
0.6936	0.6122	0.5479	0.4959	0.4286	0.3226
0.6897	0.6091	0.5455	0.4938	0.4225	0.3191
0.6857	0.6061	0.5430	0.4918	0.4167	0.3158
0.6818	0.6030	0.5405	0.4918	0.4110	0.3125
0.6780	0.6000	0.5381	0.4898	0.4054	0.3093
0.6742	0.5970	0.5357	0.4878	0.4000	0.3061
0.6704	0.5941	0.5333	0.4858	0.3947	0.3030
0.6667	0.5911	0.5310	0.4839	0.3896	0.3000
0.6630	0.5882	0.5286	0.4839	0.3846	0.2970
0.6593	0.5854	0.5263	0.4819	0.3797	0.2941
0.6557	0.5825	0.5240	0.4800	0.3750	0.2913
0.6522	0.5797	0.5217	0.4781	0.3704	0.2885
0.6486	0.5769	0.5195	0.4762	0.3659	0.2857
0.6452	0.5742	0.5172	0.4762	0.3614	0.2830
0.6417	0.5714	0.5150	0.4743	0.3571	0.2804
0.6383	0.5687	0.5128	0.4724	0.3529	0.2778
0.6349	0.5660	0.5106	0.4706	0.3488	0.2752
0.6316	0.5634	0.5085	0.4688	0.3448	0.2727
0.6283	0.5607	0.5063	0.4615	0.3409	0.2703
0.6250	0.5581	0.5042	0.4545	0.3371	

Megabits	(per second)			
0.6218	0.5556	0.5021	0.4478	0.3333
0.6186	0.5530	0.5000	0.4412	0.3297

Table 38. Supported Rates in Payload Bits Per Second: 0.26 to 0.13 Mbps

Megabits	(per second)				
0.2679	0.2222	0.1899	0.1734	0.1515	0.1345
0.2655	0.2206	0.1887	0.1724	0.1508	0.1339
0.2632	0.2190	0.1875	0.1714	0.1500	0.1333
0.2609	0.2174	0.1863	0.1705	0.1493	0.1327
0.2586	0.2158	0.1852	0.1695	0.1485	0.1322
0.2564	0.2143	0.1840	0.1685	0.1478	0.1316
0.2542	0.2128	0.1829	0.1676	0.1471	0.1310
0.2521	0.2113	0.1818	0.1667	0.1463	0.1304
0.2500	0.2098	0.1899	0.1657	0.1456	
0.2479	0.2083	0.1887	0.1648	0.1449	
0.2459	0.2069	0.1875	0.1639	0.1442	
0.2439	0.2055	0.1863	0.1630	0.1435	
0.2419	0.2041	0.1852	0.1622	0.1415	
0.2400	0.2027	0.1840	0.1613	0.1408	
0.2381	0.2013	0.1829	0.1587	0.1402	
0.2362	0.2000	0.1818	0.1579	0.1395	
0.2344	0.1987	0.1807	0.1571	0.1389	
0.2326	0.1974	0.1796	0.1563	0.1382	
0.2308	0.1961	0.1786	0.1554	0.1376	
0.2290	0.1948	0.1775	0.1546	0.1370	
0.2273	0.1935	0.1765	0.1538	0.1364	

Megabits	(per second)			
0.2256	0.1923	0.1754	0.1531	0.1357
0.2239	0.1911	0.1744	0.1523	0.1351

Table 39. Supported Rates in Payload Bits Per Second: 0.12 to 0.11 Mbps

Megabits	(per second)
0.1299	0.1195
0.1293	0.1190
0.1277	0.1186
0.1271	0.1181
0.1266	0.1176
0.1261	
0.1255	
0.1250	
0.1245	
0.1240	
0.1235	
0.1230	
0.1224	
0.1220	
0.1215	
0.1210	
0.1205	
0.1200	

Glossary [C]

asynchronous transfer mode (ATM)

A communication protocol, based on small-sized cells, *multiplexed* logical (virtual) communication channels, and fast packet switching. This protocol is especially suited for constant-bit-rate data. ATM operates over physical layer protocols (like SONET and SDH) that provide switched point-to-point connections that carry constant-bit-rate virtual channels at a wide variety of data rates. See also *ATM cell* and *virtual channel*.

ATM address

Also called ATM network address or ATM hardware address. This address is used within the ATM signaling protocol to identify an endpoint on a *switched virtual circuit (SVC)*. The address is globally unique. Two types of addresses qualify as ATM addresses: ATM NSAP and native E.164. The *ATM User-Network Interface Specification*, versions 3.0 and 3.1, specify that this address must be assigned to the endpoint by its adjacent switch, using the ILMI address registration procedure. With each *permanent virtual circuit (PVC)*, each ATM endpoint is identified by a local VC address, not by a global ATM network address. See *ILMI address registration procedure*, *ATM NSAP address*, *native E.164 address*, and *VC address*.

ATM adaptation layer (AAL)

A hardware or firmware module that converts between upper (application) layers and the ATM layer. For transmission, the AAL module creates ATM cells from the upper-layer packets, frames, or bit-stream; this function is commonly called segmentation. On reception, the AAL module converts the ATM cells into a format that is appropriate for the upper-layer application; this function is commonly called reassembly. Currently, there are 5 types of AALs: AAL1, AAL2, AAL3, AAL4, and AAL5.

ATM cell

The basic transmission unit (building block) for ATM. Each cell has 5 bytes of header and 48 bytes of payload (for example, user data or higher-layer overhead), as illustrated in Figure 32, page 220.

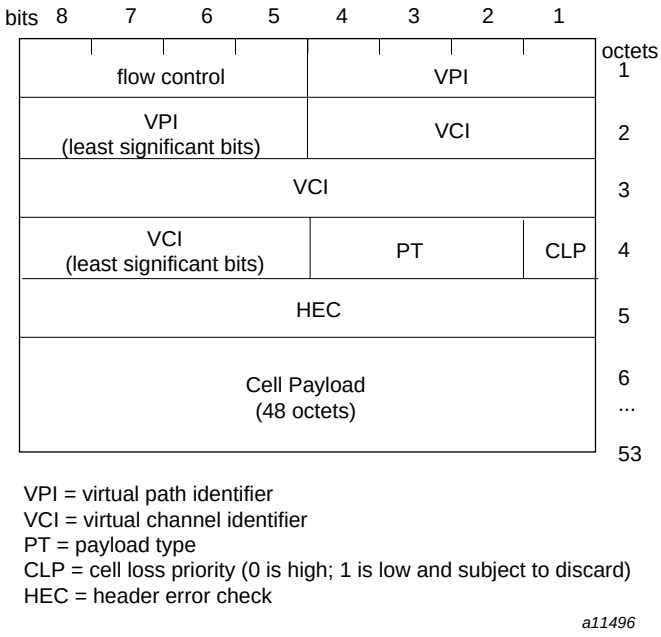


Figure 32. ATM Cell Format

ATM hardware address

The term used in RFC 1577 for ATM address. See *ATM address*.

ATM link address (VPI/VCI)

A value called in the *ATM cell* header that identifies one *virtual channel* (either permanent or switched). The address is composed of a *virtual path identifier (VPI)* and *virtual channel identifier (VCI)*. See *virtual path identifier (VPI)* and *virtual channel identifier (VCI)*.

ATM network address

A term used in ATM standards documents for ATM address. See *ATM address*.

ATM network-network interface (NNI)

A point of attachment (an interface) between two ATM devices that involves the use of the ATM NNI protocol. In general, an NNI exists at each physical connection within the boundaries of an ATM network (for example, between two ATM switches within a private ATM network). For comparison, see *ATM user-network interface (UNI)*.

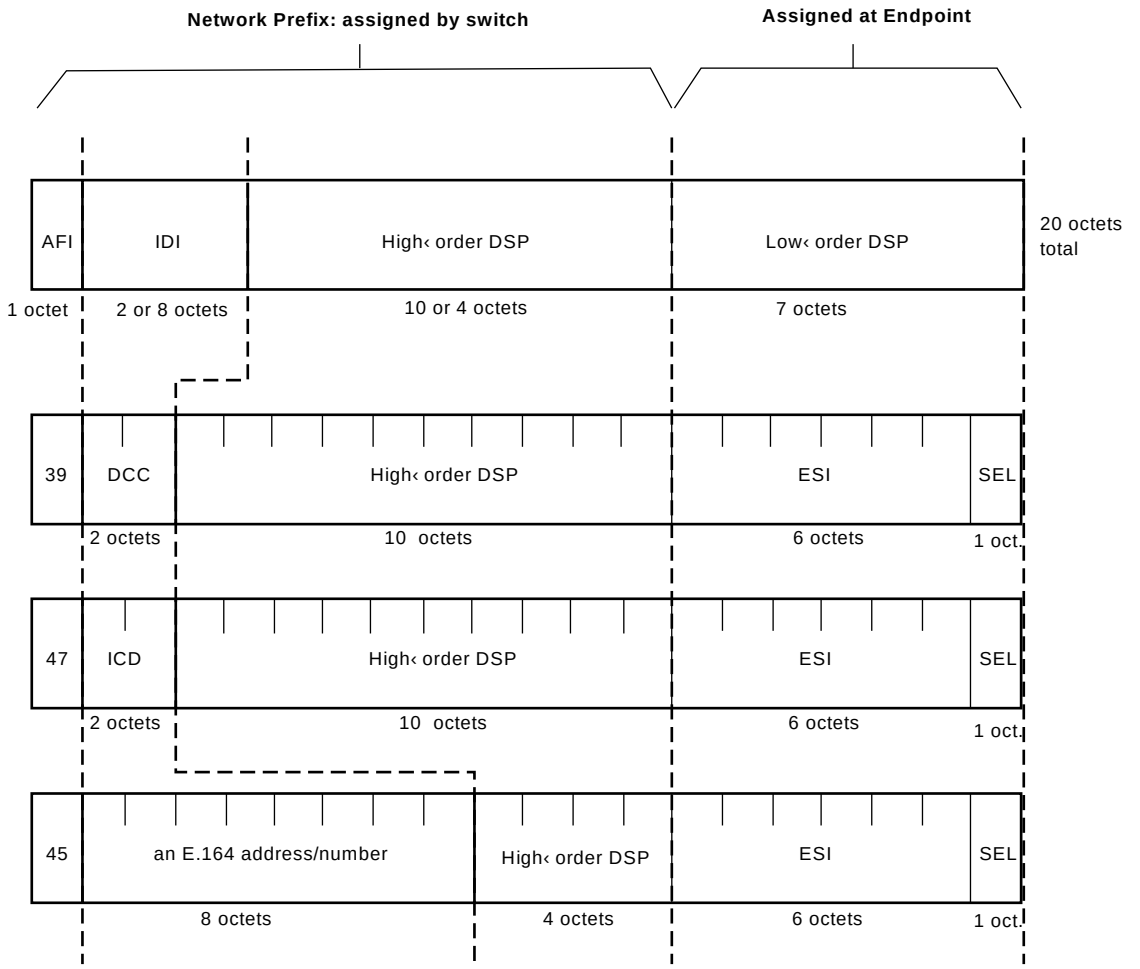
ATM NSAP address

A type of *ATM address*. The ATM NSAP address uses a three-part (that is, AFI, IDI, and DSP) format and abstract syntax that is similar to the OSI *network service access point address (OSI NSAP)*. However, ATM NSAP formats are not OSI NSAPs. All ATM NSAP addresses have a length of 20 octets, and the values for the fields are defined in the *ATM User-Network Interface Specification*. The ATM NSAP address supports three different address formats. For all formats, the AFI and IDI fields of the address are encoded in *binary coded decimal (BCD)* format. Each endpoint registers its value for the lower-order DSP portion of the address with its adjacent switch using the *ILMI address registration procedure*; the endpoint's switch assigns the value for the *network prefix* portion of the address. The format for the ATM NSAP address is illustrated in Figure 33, page 223; the valid values for the first field of the address are summarized in Table 40, page 222.

Table 40. Content for Fields of ATM NSAP Addresses

Value for AFI Field and Notation	Type of Content and Length for IDI Field	Length of DSP Field
39 decimal 0011 1001 BCD[1]	2 octets – specifying a data country code (ISO DCC value) from ISO 3166. (The 3-digit code is represented in BCD format occupying most significant 12 bits of field; least significant 4 bits are padding 1s.)	17 octets
47 decimal 0100 0101 BCD	2 octets – specifying an international code designator (ISO ICD value) from ISO 6523. (The 4-digit code is represented in BCD format.)	17 octets
45 decimal 0100 0111 BCD	8 octets – specifying an E.164 address/number up to 15-digits long. (The E.164 number is located in the least significant 60 bits of the field; most significant 4 bits of this field are always zero. Any unused bits with the 60-bit address portion of the field are padding 1s.)	11 octets

1 BCD = binary coded decimal



AFI = authority and format identifier (8 bits)
IDI = initial domain identifier (16 or 64 bits)
DSP = domain specific part (136 or 88 bits)

DCC = data country code (16 bits)
ICD = international code designator (16 bits)
ESI = end system identifier; can be a MAC address (48 bits)
IRIS ATM registers port's MAC address for this field.
SEL = end system selector; defined by local system, not by ATM standard (8 bits)
IRIS ATM software makes this field match the logical network interface number, so *atm1* uses SEL=0x01 and *atm47* uses SEL=0x2F.

a11498

Figure 33. ATM NSAP Address Formats

ATM signaling

Also called, ATM User-Network Interface (UNI) signaling. (This definition does not describe ATM Network-Network Interface (NNI) signaling.) A protocol used between an endpoint and its adjacent switch, whose purpose is to manage and set up a *switched virtual circuit (SVC)* in real-time. ATM signaling communication occurs on a *permanent virtual circuit (PVC)* using the VPI/VCI value of 0/5. The protocol specifies a set of communications that allow the following sequence of events to take place:

- A calling endpoint requests that a *virtual channel* be set up to another endpoint (the called endpoint) and specifies a *traffic contract* for the channel. This request is sent to the attached (adjacent) ATM switch using the user-to-network interface (UNI).
- The ATM switch determines a route to the called endpoint and sets up a connection. It then passes the request either to the called party or to the next switch enroute to the called endpoint.
- Each switch along the route does the same function as the first switch.
- When the called party receives the setup request through its UNI, it responds with an acceptance or a rejection.
- When the connection is accepted and the connection message has reached the calling party, the two endpoints exchange data according to the traffic contract.
- When the connection is rejected, the connection is torn down all along the route as the rejection message makes its way back to the calling endpoint.

Signaling is specified by the *ATM User-Network Interface Specification* and the ITU-T Q.2391 standard; it functions as a layer 3 entity. The protocol is used ("spoken") over one publicly-defined (that is, well-known) *permanent virtual circuit (PVC)* in order to set up a system's switched virtual channels. Among other things, the protocol allows endpoints to open and close connections, negotiate *traffic contract*s, and add and remove parties from multicast connections.

ATM user-network interface (UNI)

A point of attachment (an interface) between two ATM devices that involves use of the ATM UNI protocol. In general, a UNI exists at each point of entry into an ATM network (for example, between an endpoint and a switch or between a private ATM switch and a public ATM switch). For comparison, see *ATM network-network interface (NNI)*.

available bit rate (ABR)

A characteristic of a *virtual channel* whereby cells are placed on the physical medium and managed during their transport in a manner that uses bandwidth that is not being used for *constant bit rate (CBR)* or *variable bit rate (VBR)* traffic. ABR is intended for bursty traffic (for example, electronic mail or file transfers).

bearer class

See *transport class*.

binary coded decimal (BCD)

A notation for representing decimal numerals 0 to 9 using 4-bit binary sequences. The valid binary sequences range from 0000 (for decimal zero) to 1001 (for decimal nine). Each octet (8-bit byte) represents two decimal numerals. For example, the octet 0100 0111 represents 47 decimal. For comparison, 47 decimal in standard binary format, is represented with the following binary sequence: 0010 1111.

burst tolerance

The maximum number of cells that a service user is allowed to transmit at the virtual channel's *peak cell rate (PCR)*. Said another way, the longest burst an application can make on a *variable bit rate (VBR)* channel. This is one component of a variable bit rate VC's *traffic contract*. See also *peak cell rate (PCR)* and *sustainable cell rate*.

cell delay variation (CDV)

Also known as jitter. A performance criteria that measures how consistently ATM cells on a *virtual channel* arrive at the receiving endstation. For example, if a VC's traffic contract specifies that cells arrive every 110 microseconds, the CDV for

that connection increases every time a cell arrives either earlier or later than expected. For constant bit rate traffic, even small CDV values can affect the perceived quality of the data transfer. CDV can be affected by variation in the any of the stations that transmit the cell along the entire connection.

This error is caused when two or more cells (from different VCs) need to be placed into the exact same timing slot within a transmission stream in order to conform to the rates for those VCs. The error occurs on each cell that is not placed in its intended slot.

One-point CDV compares the variability in arrival times in reference to the negotiated *peak cell rate (PCR)*. Two-point CDV measures the variability in arrival times in reference to the time of transmission.

cell error ratio

A measurement of the accuracy of ATM data transfer. The result of dividing the number of ATM cells that contain errors by the total number of ATM cells processed. An error can be an invalid ATM header or corrupted payload.

cell interarrival variation (CIV)

See *cell delay variation (CDV)*.

cell loss priority (CLP)

A 1-bit level of importance indication, carried within the header of the *ATM cell*. A CLP set to 1 indicates that the cell can be discarded if the network experiences congestion. A CLP set to 0 indicates that the cell should only be discarded as a last resort.

cell loss ratio

A measurement of the dependability of ATM data transfer. The result of dividing the number of lost ATM cells by the total number of transmitted cells. A lost cell is one that was transmitted but not received within an acceptable period of time.

cell misinsertion rate

A measurement of the accuracy of ATM data transfer. The result of dividing the number of misinserted cells by a time

interval. A misinserted cell is a cell received on a VC for which there is no corresponding transmitted cell.

cell transfer delay

A measurement of the speed of ATM data transfer. Cell transfer delay is the elapsed time between when a cell is transmitted and when it is received.

constant bit rate (CBR)

A characteristic of a *virtual channel* whereby cells are placed on the physical medium and managed during their transport in such a precise manner that each cell arrives at its destination "by the clock". The time delay between the arrivals of the cells is as close to equal (that is, constant) as physically possible. A CBR channel carries data at its *peak cell rate (PCR)* all the time. This corresponds to ATM quality of service class A (or 1) and is intended for use by applications such as video-on-demand and telephone conversations. Compare to *variable bit rate (VBR)*.

E.164 address

See *native E.164 address* and *ATM NSAP address*.

embedded transport rate

The number of bits of information that a logical channel carries every second. An embedded transport rate must be lower than or equal to the *signal rate* of the physical layer.

encapsulation

Also called overlayering. A situation where one network protocol is embedded (encapsulated) within another network protocol's logical structure so that the embedded protocol can be carried transparently (tunneled) through the other network.

Figure 34, page 228, illustrates the general header format for an encapsulated packet.

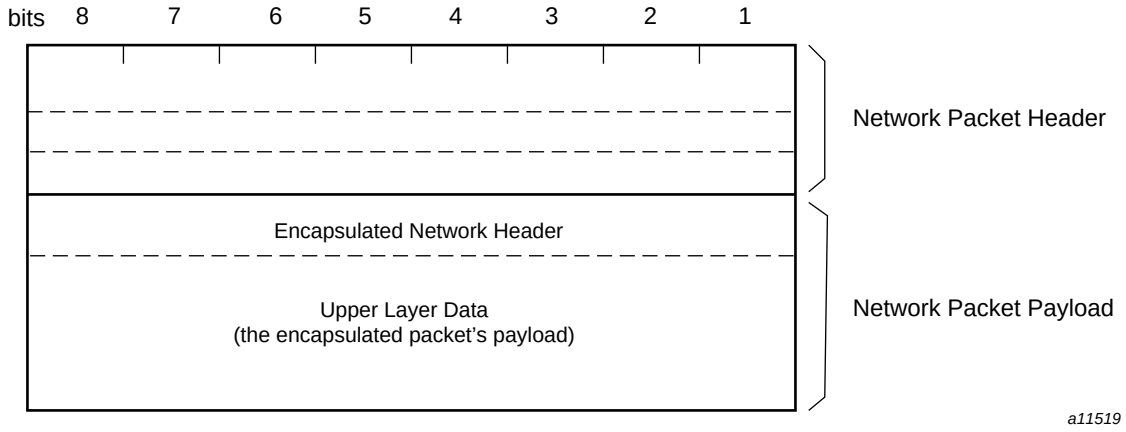


Figure 34. Encapsulation

endpoint

The point at which an ATM service user (original source or ultimate destination) passes the ATM cell payload to the ATM layer or accepts the ATM cell payload from the ATM layer.

ILMI address registration procedure

A protocol defined in the *ATM User-Network Interface Specification*, versions 3.0 and 3.1. The protocol is used by an endpoint and its adjacent switch to assign and register a unique *ATM address* to the endpoint.

interim local management interface (ILMI)

An ATM-related protocol that provides status, configuration, address registration, and control information about link and physical layer parameters for an ATM user-network interface. The primary purpose of ILMI is to dynamically discover and register each UNI's ATM address.

International Telecommunications Union-Telecommunication Sector

Formerly known as CCITT.

IP-to-ATM address resolution table

A database (for example, a lookup table) that maps IP addresses to ATM addresses. For PVC environments, the table maps IP addresses to VPI/VCI/port tuplets; each endpoint maintains its own table and does its own address resolution. For SVC environments, the table maps IP addresses to ATM NSAP or native E.164 addresses. For RFC 1577-compliant environments using SVCs, the ATMARP server for each IP logical subnet (LIS) maintains this table and responds to requests for address resolution from LIS members. See *ATM NSAP address*, *native E.164 address*, and *VC address*.

ITU-T

See International Telecommunications Union-Telecommunication Sector.

jitter

See *cell delay variation (CDV)*.

line rate

See *signal rate*.

link

A point-to-point physical communication medium. For example, the cable connecting a transmitting station to a switch (or a hub or concentrator) is one link of a longer connection that requires at least one more link in order to reach the receiving station. See also *virtual channel link* and *virtual path link*.

logical IP subnetwork (LIS)

A collection of ATM endpoints that all share the same IP network address and mask (that is, the same subnet address) and that can all contact each other directly through the ATM network. The term and its definition are part of the RFC 1577 design for doing IP-over-ATM.

mean cell transfer delay

A measurement of the speed of ATM data transfer. An arithmetic average of a specified number of *cell transfer delay*s for one or more connections.

multiplex

To make one entity serve multiple purposes. In communication environments, this means transmitting multiple communication streams over a single physical medium. For example, 3 conversations can be carried over a single wire by using a different signal frequency for each conversation (an analog multiplexing solution) or by interleaving bytes from the conversations (a digital multiplexing solution).

native E.164 address

Also called non-NSAP E.164. A type of *ATM address*. The format for the native E.164 address is defined in *CCITT Recommendation E.164*. The addresses are administered and assigned by public networks (for example, telephone companies). This address can be up to 15-bytes in length, as illustrated in Figure 35, page 230.

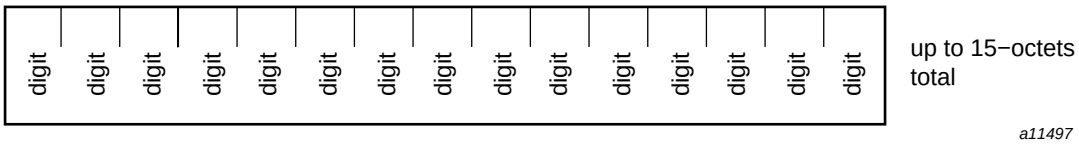


Figure 35. E.164 Address Format

network prefix

For ATM NSAP addresses, the network prefix is that portion of the address that is assigned by the ATM switch. This includes all fields in the address except the ESI and SEL fields.

network service access point address (OSI NSAP)

A variable-length address that identifies the location (for example, software module) where an OSI transport layer (Layer 4) accesses the services of an OSI network layer (Layer 3). The NSAP address is defined by OSI documents ISO 8348 and ISO 10589, where values for the different fields are specified and explained. The OSI NSAP format is illustrated in Figure 36,

page 231; the valid values for the first field of the address are summarized in Table 41, page 231.

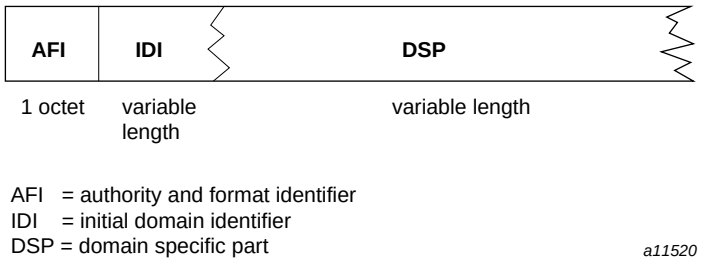


Figure 36. OSI NSAP Format

Table 41. Meanings for Values in NSAP Fields

AFI (1 octet)	Contents of IDI
37	X.121
39	ISO DCC (data country codes from ISO 3166)
41	F.69 (telex)
43	E.163 (public switched telephones)
45	E.164 (15-digit ISDN numbers)
47	ISO ICD (international code designators from ISO 6523)
49	Local: for use by private organizations
50	Local: ISO 646 characters
51	Local: national characters

The authority and format identifier (AFI) field identifies the organization that is authorized to allocate/assign the values used in the subsequent initial domain identifier (IDI) field. The AFI value determines the format and length of the IDI field. Likewise, the IDI field identifies the format and length of the domain specific part (DSP) field.

Within an ATM context, this basic NSAP format is used, however, the usage is different. See also *ATM NSAP address*.

NSAP

See *network service access point address (OSI NSAP)*.

peak cell rate (PCR)

The maximum transmission rate possible on a *virtual channel* (VC). For a constant bit rate VC, this is the rate at which the service user transmits data. For a *variable bit rate (VBR)*, this value is the maximum rate that a bursty service user is likely to use. Peak cell rate is one component of each VC's *traffic contract*.

operation and maintenance (OAM)

Functions that exist at various SONET and ATM levels for the purpose of collecting and reporting status that is useful for operating and maintaining a SONET/ATM network.

performance parameter

A measurable characteristic of a communication service. Every protocol uses different performance parameters to define and measure its performance. For the ATM protocol, performance parameters are specified in order to define the *quality of service (QoS)* classes, as well as to describe and measure performance on a connection. The most important ATM performance parameters are listed below.

- peak-to-peak *cell delay variation (CDV)*
- maximum *cell transfer delay*
- *mean cell transfer delay*
- *cell loss ratio*

Other ATM performance parameters include the following:

- *cell error ratio* for *cell loss priority (CLP)* level 0 for the VC
- cell error ratio for cell loss priority level 1 for the VC
- *severely-errored cell block ratio*
- *cell misinsertion rate*
- *mean cell transfer delay*

- definition for cell conformance
- definition for connection compliance

See also *quality of service (QoS)* and the separate entry for each ATM performance parameter.

permanent virtual circuit (PVC)

A long-lived *virtual channel* that is established through a service order or pre-arranged network management. The *traffic contract* for this type of channel is negotiated before the service is purchased/installed, and does not change easily, hence the term "permanent." If any section of a PVC connection fails while the PVC is active, the connection is automatically reopened once repairs have been completed. Contrast this with the *switched virtual circuit (SVC)*.

quality of service (QoS)

In general, a specified set of *performance parameter* objectives that define one level (class) of service. Different parameters and/or objectives for the parameters are grouped into a number of different classes. Customers select the class that fits their needs. As an analogy, for the protocol referred to as overland mail carrier (or U.S. Postal Service), the QoS classes include First, Second, Third, Registered, Certified, and Bulk. Some of the parameters that define these classes are number of days to delivery, level of insurance against content loss or damage, and proof of delivery via a signature at the destination.

In the ATM environment, each *virtual channel* (VC) has one QoS class associated with it as part of its *traffic contract*. For a *switched virtual circuit (SVC)*, there is a QoS for each direction (one for forward; one for back/return). A group of VCs that have different QoS classes can be carried together in one *virtual path connection (VPC)*; however, the VPC must meet the requirements for the most demanding QoS among the carried VCs.

The five currently defined ATM QoS classes are listed below. At present, performance parameter objectives for each ATM QoS class vary from provider to provider, and even from contract to contract.

- QoS Class 0 = Unspecified QoS (for example, best effort)

- QoS Class 1 = Circuit emulation (for example, constant bit rate video)
- QoS Class 2 = Variable bit rate audio and video
- QoS Class 3 = Connection-oriented data transfer
- QoS Class 4 = Connectionless data transfer

See also *performance parameter* and *traffic contract*.

rate management (RM)

A flow-control protocol associated with the ABR service class.

severely-errored cell block ratio

A measurement of the accuracy of ATM data transfer. The result of dividing the number of ATM cell blocks that contain any of various serious errors by the total number of ATM cell blocks processed. A "block" consists of consecutively transmitted ATM cells that lie between successive OAM cells on a *virtual channel connection (VCC)*.

signal rate

Also called line rate. The number of bits of information that a physical layer protocol carries every second across the entire length of its physical medium. In synchronous optic network (SONET) environments, the physical medium is a length of fiber-optic cable connecting a SONET transmitter and a receiver. SONET signal rates are usually expressed in megabits per second. See *synchronous optic network (SONET)*.

signaling ATM adaptation layer (SAAL)

An ATM adaptation layer (AAL) that supports signaling. See also *ATM adaptation layer (AAL)* and *ATM signaling*.

SONET

See *synchronous optic network (SONET)*.

SONET frame

Also called STS-1 frame. The basic logical transmission unit used by the SONET protocol. Each SONET frame represents data for one 51.84 megabit per second synchronous transport signal (STS-1). The frame is illustrated in Figure 37, page 235. All SONET signal rates are multiples of the STS-1 rate, and the SONET signal is created by interleaving (multiplexing) bytes from SONET frames. For example, a SONET OC3 (155.52 megabits-per-second) connection carries 3 different STS-1 signals, each using SONET frames to encapsulate its data. The OC3 signal is composed by taking, in turn, the first byte from each signal's first frame, then the second byte from those frames, until all three frames are transmitted, and continuing this process for subsequent frames.

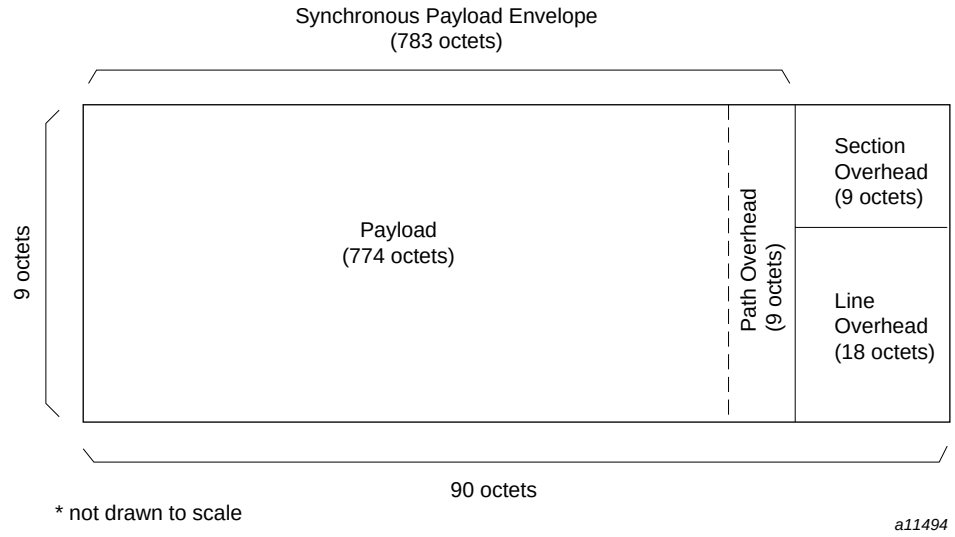


Figure 37. SONET Frame Format

sustainable cell rate

The average transmission rate that a *variable bit rate (VBR)* service user is likely to use on a *virtual channel*. This is one component of a variable bit rate VC's *traffic contract*. See also *peak cell rate (PCR)* and *burst tolerance*.

switched virtual circuit (SVC)

A short-lived, by-demand *virtual channel* that is established through software negotiation (that is, through *ATM signaling*). The *traffic contract* for this type of channel is negotiated at the time the channel is requested. If any section of an SVC connection fails while the SVC is active, the connection is not automatically reopened; the parties involved time out, then reinitiate the connection. Contrast this with *permanent virtual circuit (PVC)*.

synchronous optic network (SONET)

An American National Standard (ANSI) protocol for high-speed optical telecommunications. This physical layer protocol is based on optic fiber and synchronous digital multiplexing. SONET supports a wide variety of *line rate*s, starting at 51.84 megabits per second and increasing in multiples of this base rate (for example, 155.52 and 622.08).

synchronous payload envelope (SPE)

The portion of a *SONET frame* that is provided by the SONET path layer. This area consists of 9 octets of path overhead and 774 octets of payload.

traffic contract

A set of negotiated performance objectives associated with a *virtual channel* (VC). These objectives are negotiated between a service user and a service provider. The negotiation can be done in any one of three methods: (1) in real time, either through *ATM signaling* or through a network management system, (2) prior to installation through a verbal or written agreement, or (3) by default (that is, the user accepts whatever performance comes with the service). Real time negotiation via signaling sets up the *switched virtual circuit (SVC)*. The other types of negotiation set up a *permanent virtual circuit (PVC)*.

For private service providers, the traffic contract may consist of any set of parameters.

For public service providers, each traffic contract must include at least the following items:

- A *quality of service (QoS)* class, for each VC (for example, the forward VC and, if one exists, the backward VC), that specifies performance criteria for at least the allowed *cell delay variation (CDV)*, and definitions for connection compliance and cell conformance.
- The *performance parameter*s: for example, a *peak cell rate (PCR)* for each *cell loss priority (CLP)* level used on the VC.

In addition, the contract may include performance objectives for the following traffic parameters. Not all of these parameters can be specified for all service classes.

- A *sustainable cell rate*
- A *burst tolerance*
- A minimum cell rate
- A best effort indication
- Allowed levels for QoS *performance parameter*s

See also *quality of service (QoS)* and *performance parameter*.

transport class

Also known as bearer class. In general, levels of service provided by the entities that transport (that is, bear or carry) the data. These entities are usually referred to as a network. For ATM, there are currently three transport classes, as described below. The exact performance of each class varies from provider to provider because the *traffic contract* is a negotiated item that can be implemented in more than one manner.

- Transport Class X: connection oriented. User specifies all other traffic contract parameters.
- Transport Class A: connection oriented, constant bit rate with end-to-end timing requirements and stringent cell loss, cell delay, and cell delay variation performance. User specifies only bandwidth (bit rate) and quality of service (QoS).
- Transport Class C: connection oriented, variable bit rate. No end-to-end timing requirements. User specifies only bandwidth (bit rate) and QoS.

See also *quality of service (QoS)* and *traffic contract*.

UNI

See *ATM user-network interface (UNI)*.

unspecified bit rate (UBR)

A characteristic of a virtual channel whereby cells are placed on the physical medium and managed during their transport in a manner that does not require conformance to a *traffic contract*. Each network provider may define locally a set of *performance parameter* objectives for UBR, but these do not have to remain constant during the duration of the connection, nor across all links making up the connection. UBR is, by definition, a best-effort transmission, and does not guarantee delivery of the data. This corresponds to ATM quality of service class 0.

variable bit rate (VBR)

A characteristic of a *virtual channel* whereby cells are placed on the physical medium and managed during their transport in a manner that allows the delay between the arrivals of cells to be unequal (that is, to be variable) even while the bandwidth is guaranteed. The *cell delay variation (CDV)* remains within the limit specified by the virtual channel's *traffic contract*. VBR virtual channels are appropriate for bursty transmitters. VBR channels include *peak cell rate (PCR)*, *sustainable cell rate*, and *burst tolerance* in their traffic contracts. This corresponds to ATM quality of service class B and C (or 2 and 3) that is intended for applications such as Frame Relay internetworking and compressed video. Compare to *constant bit rate (CBR)*.

VC address

The address that uniquely identifies one *virtual channel*. This address consists of a *virtual path identifier (VPI)*, *virtual channel identifier (VCI)*, and an ATM hardware port identifier. The address is valid only at the ATM station that creates the address. The VC address for each link along an end-to-end connection is different.

virtual channel

Also called logical connection and virtual circuit. A logical communication stream that provides sequential, unidirectional

transport of ATM cells in accordance with a *traffic contract* that is known to both the sender and the receiver. Each virtual channel (VC) is for one conversation, video stream, or other endpoint-to-endpoint communication. Each VC is identified by an address value carried in the header of the *ATM cell*: this value is the concatenation of the *virtual path identifier (VPI)* and *virtual channel identifier (VCI)*. VCs come in two varieties—permanent and switched—depending on how their performance parameters (traffic contracts) are negotiated. See also *permanent virtual circuit (PVC)* and *switched virtual circuit (SVC)*.

virtual channel connection (VCC)

A concatenation of *virtual channel link*s that carry the data for a *virtual channel*. A VCC extends from one endpoint to another; it begins and terminates where the two ATM service users (sender and receiver) access the ATM layer. The collection of physical entities involved in carrying a virtual channel.

virtual channel identifier (VCI)

A 1- to 16-bit address that in combination with the *virtual path identifier (VPI)* identifies an active *virtual channel*. Any specific VCI value is meaningful only along one *virtual channel link* of the *virtual channel connection (VCC)*; the VCI value in the *ATM cell*'s header is reassigned (and overwritten) at each switch.

virtual channel link

A unidirectional physical medium that transports ATM cells between a point where a channel address (that is, VCI) is assigned and a point where the value is translated or removed.

virtual path (VP)

A collection of *virtual channel*s carried together in a single multiplexed stream, all destined for the same terminator (for example, switch). The VP is identified by a *virtual path identifier (VPI)*.

virtual path connection (VPC)

A concatenation of physical *virtual path link*s that extends between two *virtual path terminator*s. The connection carries a virtual path (that is, a bundle of virtual channel). Each VPC has

a virtual path identifier associated with it. See also, *virtual path link*, *virtual path terminator*, and *virtual path identifier (VPI)*.

virtual path identifier (VPI)

A 1- to 8-bit address that identifies an active *virtual path (VP)*. Any specific VPI value is meaningful only along one *virtual path link* of the *virtual path connection (VPC)*; the VPI value in the *ATM cell*'s header is reassigned (and overwritten) at each *virtual path terminator*.

virtual path link

A unidirectional physical medium that transports ATM cells between a point where a path address (that is, VPI) is assigned and a point were the value is translated or removed.

virtual path terminator

A node or system within an ATM network that unbundles the *virtual channel*s contained within a *virtual path (VP)* and processes each VC independently, which may include rebundling VCs into new virtual paths.

well-known virtual channel

A virtual channel that is reserved for a specific purpose, such as signaling traffic or ILMI communications. The *ATM User-Network Interface* standard reserves the VCI values (within each VPI) from 0 to 31 for use as well-known channels. The reserved well-known VPI/VCI for ATM signaling traffic is specified as 0/5, and for ILMI communications it is 0/16.

Index

Tell Us About This Manual

As a user of Silicon Graphics products, you can help us to better understand your needs and to improve the quality of our documentation.

Any information that you provide will be useful. Here is a list of suggested topics:

- General impression of the document
- Omission of material that you expected to find
- Technical errors
- Relevance of the material to the job you had to do
- Quality of the printing and binding

Please send the title and part number of the document with your comments. The part number for this document is 007-2333-005.

Thank you!

Three Ways to Reach Us

- To send your comments by **electronic mail**, use either of these addresses:
  - On the Internet: techpubs@sgi.com
  - For UUCP mail (through any backbone site): *[your_site]*!sgi!techpubs
- To **fax** your comments (or annotated copies of manual pages), use this fax number: 650-932-0801
- To send your comments by **traditional mail**, use this address:

  Technical Publications
  Silicon Graphics, Inc.
  2011 North Shoreline Boulevard, M/S 535
  Mountain View, California 94043-1389