ESV Workstation

LAT Host Services User's Manual

EVANS & SUTHERLAND COMPUTER CORPORATION
Salt Lake City, Utah

DOCUMENTATION WARRANTY:

PURPOSE: This documentation is provided to assist an Evans & Sutherland trained BUYER in using a product purchased from Evans & Sutherland. It may contain errors or omissions that only a trained individual may recognize. Changes may have occurred to the hardware/software, to which this documentation refers, which are not included in this documentation or may be on a separate errata sheet. Use of this documentation in such changed hardware/software could result in damage to hardware/software. User assumes full responsibility of all such results of the use of this data.

WARRANTY: This document is provided, and Buyer accepts such documentation, "AS-IS" and with "ALL FAULTS, ERRORS, AND OMISSIONS." BUYER HEREBY WAIVES ALL IMPLIED AND OTHER WARRANTIES, GUARANTIES, CONDITIONS OR LIABILITIES, EXPRESSED OR IMPLIED ARISING BY LAW OR OTHERWISE, INCLUDING, WITHOUT LIMITATIONS, ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. BUYER FURTHER HOLDS SELLER HARMLESS OF ANY DIRECT OR INDIRECT DAMAGES, INCLUDING CONSEQUENTIAL DAMAGES.

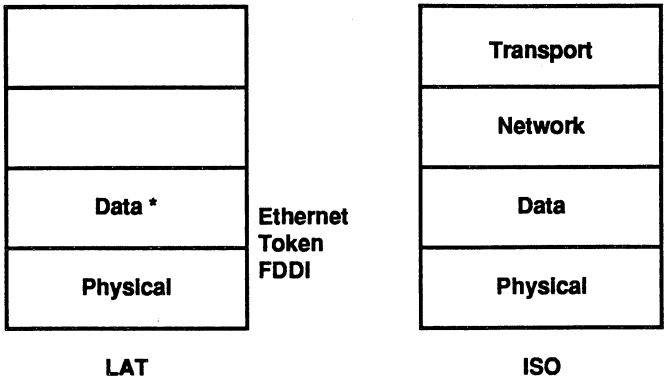
Part Number: 517941-601 AA
April, 1991

Table of Contents

1. Overview

LAT Host Services Components

Local Area Transport (LAT) is a network protocol developed primarily by Digital Equipment Corporation. The diagram below details the LAT protocol layer and ISO Open System Interconnect model.



The LAT protocol provides transport layers which enable remote login to the host from another LAT terminal device and print services from the host to a remote LAT printer. Current implementations run using an Ethernet data/physical link, but could easily run on an equally fast medium such as a 4-16 megabit token ring or a 100 megabit Fiber Distributed Data Interface (FDDI). The network layer, or virtual circuit layer, establishes and maintains a virtual circuit between a services node and a server.

The transport layer, or slot layer, establishes service sessions and the exchange of data by multiplexing and demultiplexing data transferred between a server port and service session on the service or host node.

The LAT Host Services components (**lat** and **latcp**) and your system use the LAT protocol to provide:

1) Login access to your system from terminals connected to terminal servers or workstations/PCs appearing as terminals connected to a terminal server.
2) System access to printer(s) connected to a terminal server.
3) Ability to gather LAT network statistics.
4) Tools to clearly define problems if and when they arise.

lat

An event driven protocol engine and the following three interfaces comprise the **lat** component: *datalink* interface, *command/management* interface, and *multiple user_services* interfaces.

The heart of the **lat** component is the event driven protocol engine. It has two states, idle (wait) state and process event state. In the idle state, the protocol engine waits until an event is present. As events are queued by each of the three interfaces into their respective event queues, the protocol engine processes them.

The protocol engine removes a single event from the datalink interface's event queue first, the command/management interface's event queue second and each of the multiple user_service interface's event queues third. It repeats the process until no queue contains any events. It then enters the idle state. Each event is processed to completion at each queue before proceeding to the next queue in the above specified order.

The following pseudo code serves to further clarify the event processing logic.

```
dowhile (event present)
      if (datalink interface's event present) then
            process (next datalink interface event)
      endif
      if (command/management interface event present) then
            process (next command/management interface event)
      endif
      for (each of the user_service interface's queues)
            if (user_service interface's event present for
                  port[number]present) then
                  process (next user_service interface event)
            endif
      endfor
enddowhile
```

Table 1-1 lists typical events associated with each interface and the action taken as a result of the event.

Table 1-1. Interface event table

Interface	Event	Action Taken
datalink	Receipt of 6004 protocol type network packet	Process **lat** messages including starting and maintaining **lat** demultiplexing data in lat slots coming from a terminal server port.
management/ command	Request to display **lat** statistics.	Reads **lat** counters in virtual circuit slot and executor databases and displays.
management/ command	Ask server to make a connection from the specified port.	Builds host initiated message and sends it to server.
management/ command	Start protocol trace.	Writes all packets to user specified disk file for later analysis.
management/ command	Stop protocol trace.	Closes disk file containing **lat** protocol messages in binary format.
management/ command	Stop **lat** program.	Complete rundown of **lat** program includes display of final network statistics.
management/ command	Request to turn on debugging.	Depending on the level of debug specified, the **lat** process will display **lat** events and messages.
user_services interface	Characters/data from print spoolers or terminal input/output.	Places characters/data in slot messages and transmits to terminal server.

The datalink interface forwards received network packets with protocol type 6004 to the **lat** protocol engine. It serves as an interface for transmitting **lat** protocol messages.

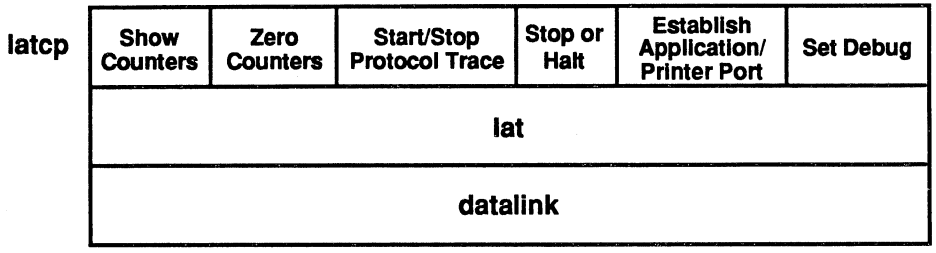
The command and management interface provides input from user commands entered via a **lat** command line or **latcp** command line interface.

The user service interface provides data from user services such as user process via **/bin/login**.

latcp

This is a program which communicates management and command requests to a **lat** component protocol engine. **latcp** places requests into the command management event queue. These requests include the following:

1) Zero slot, circuit counters
2) Show executor, circuit, and slot counters
3) Stop **lat** or **latcp**
4) Establish an application port for use as a local system printer port
5) Set delay mode for display of internal info
6) Turn protocol tracing on and off



The command syntax and returned value are detailed in chapter 5, "LAT Statistics." Set debug and protocol trace are detailed in chapter 6, "Troubleshooting LAT."

lat, latcp and Your System's Relationship

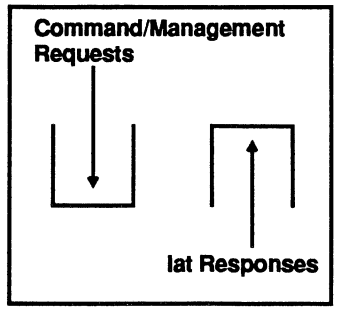
lat and **latcp** are executable programs which run in *user process space* and communicate with the Ethernet.

lat is forked as a background task when started. It may be started with a **-nofork** option and run in the foreground.

```
USER      PID   SZ    RSS   TTY    STAT  TIME  COMMAND
corrigan  651   96    48    pad00  S     0:00  su10
root      652   160   122   pad00  S     0:01  sh
root      654   160   133   pad00  S     0:00  lat -nofork
```

In the above example, **lat** is running in user space with a LAT command line interface.

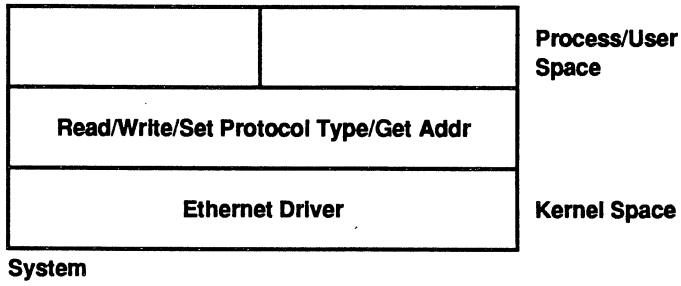
From the LAT interface, or from **latcp** if the **lat** is forked, you can gather statistics, set up printer ports, and do troubleshooting.



lat responds by displaying the requested statistics, attempting to establish a printer port, and tracing the **lat** protocol stack or displaying debug info.

Pseudo terminals are software terminals. There is no corresponding hardware associated with a pseudo terminal. Print spoolers generate output to devices. These devices may be pseudo devices such as pseudo terminals. LAT uses pseudo terminals for logging in remote users and for connecting remote devices. With a print spooler output directed to a pseudo terminal that LAT has connected to a remote device, the printing is done on this remote device.

In addition to the pseudo terminal and printer port interfaces, there is the datalink interface. The current datalink interface is an Ethernet interface. Packets with protocol type 6004 are received from the network for processing. Packets bound for terminal server ports, and miscellaneous keepalive and service messages, are transmitted over the datalink interface.



The datalink code is in the kernel with a user interface. These Ethernet routines are not generally available to the user but are available to the **lat** program.

2. Starting lat

This chapter explains how **lat** is automatically started and how to start **lat** from the command line.

Automatic lat Startup

When the system is powered up in multi-user mode, the file **/etc/init.d/lat** is executed with **start** as a parameter. This script starts **lat** and runs it as a daemon.

Starting lat from the Command Line

Forked

You can simply enter the **lat** command, and **lat** will be forked, displaying output similar to the following:

```
Ki2: lat
Ki Research, Inc: LAT Version 4.2.5

Forking Network Daemon.
```

This command is usually placed in the startup command file.

Not Forked

You can also create a command line interface to **lat** by specifying the **-nofork** option. The following output will be displayed if you enter the **lat -nofork** command:

```
Ki2: lat -nofork
Ki Research, Inc: LAT Version 4.2.5

LAT Rating File: '/usr/etc/rating.lat'

Default Ethernet name is la0
Max Session Allowed:  16
```

The lat Options

Table 2-1 lists each **lat** option by name, describes it and gives an instance of when you may want to use it.

Table 2-1. lat options

Option	Description
-banner=	This text is displayed when a user connects to **lat**. The default is *Welcome to Ki Research, Inc. LAT Host Services*. Use if you want to specify a banner.
-[no]fork	Run **lat** as a background daemon. (Default: **fork**) If you want to run **lat** in foreground: **-nofork**. If you want to run **lat** in background: **-fork**.
-groups=#	Set the **lat** Groups to which this node belongs. The default is Group 0. Valid groups are 0 to 255. A range of groups may be specified as **#-#**; *i.e.*, 0-5. Use if you want more than/other than Group 0.
-id=	The local node identification. By default this is set to *Ki LAT Host Services*. Use if you want to specify an ID.
-lan=	The LAN identifier, such a **le0**, **en0**, or **0**, for non-UNIX systems. Use if you have an alternate LAN device.
-rating=#	The rating to assign the local node. By default this value is 6. Use if you want a value other than 6.
-service=	The service name to declare. The default is the local hostname. Any number of service names may be declared. Use if you want to specify a service.
-service_id=	The service identification. The default value is *Interactive Login*. Each ID matches any specified services in the order in which they are specified. Use if you want to specify a service ID.
-sessions=#	Allowed number of sessions. The maximum is 1024. Use if you want to restrict access.
-[no]uppercase	Use lower case node and service names (default: **uppercase**). If you have terminal servers that can handle lower case names use **-uppercase**. If you have terminal servers than can't handle lower case names use **-nouppercase**.

3. Logging In from Your Terminal Server

The total number of LAT terminal server ports has reached roughly two million (2,000,000) Ethernet terminal ports. This is a sizeable number, repeating a well-established and useful terminal access method.

To access your system from a terminal server, use the **connect *<nodename>*** command, where ***nodename*** is usually the system's hostname. A different node name may have been specified when starting **lat**. For example,

```
> connect ims
```

You can show the available services/nodes/hosts that you can connect to by entering the **show services** or **show node** commands. For example,

```
> show nodes
                Summary Display
NODE NAME          STATUS          IDENTIFICATION

(name of node)     REACHABLE       (word or phrase)
                   UNREACHABLE
                   UNKNOWN
                   {n}CONNECTED


> show services
             Service Summary Display
NODE NAME           STATUS          IDENTIFICATION
(name of service)   REACHABLE       (word or phrase)
                    UNREACHABLE
                    UNKNOWN
                    {n}CONNECTED
```

4. Setting Up Print Services

This chapter contains the information necessary to set up a printer port on the terminal server for your host's use. It gives suggested modifications to your **printcap**.

Establishing a Printer Port

In order to establish a printer port on the terminal server, you must know the server's name and the server's port_name. In order to determine the terminal server's name, enter the **show server** command on a **lat** terminal attached to the terminal server. For example,

```
>show server
      Server Characteristics Display
VISTA 1.0   VERSION:       UPTIME:
ADDRESS:          NAME:       NUMBER:
CIRCUIT TIMER:                PASSWORD LIMIT:
CONSOLE PORT:                 QUEUE LIMIT:
INACTIVITY TIMER:             RETRANSMIT LIMIT:
KEEPALIVE TIMER:              SESSION LIMIT:
MULTICAST TIMER:              SOFTWARE:
NODE LIMIT:
MAXIMUM CIRCUITS:
MAXIMUM SERVICES

SERVICE GROUPS:
ENABLED CHARACTERISTICS:
```

To determine the port's name, enter the **show port** command. For example,

```
>show port
      Show Port Status Display
PORT {number}:    {port user name}
ACCESS:                 CURRENT SERVICE:
STATUS:                 CURRENT MODE:
SESSIONS:               CURRENT PORT:
INPUT XOFFed:           OUTPUT SIGNALS:
OUTPUT XOFFed:          INPUT SIGNALS:
```

The default port name is **port_*n***, where ***n*** is the port number.

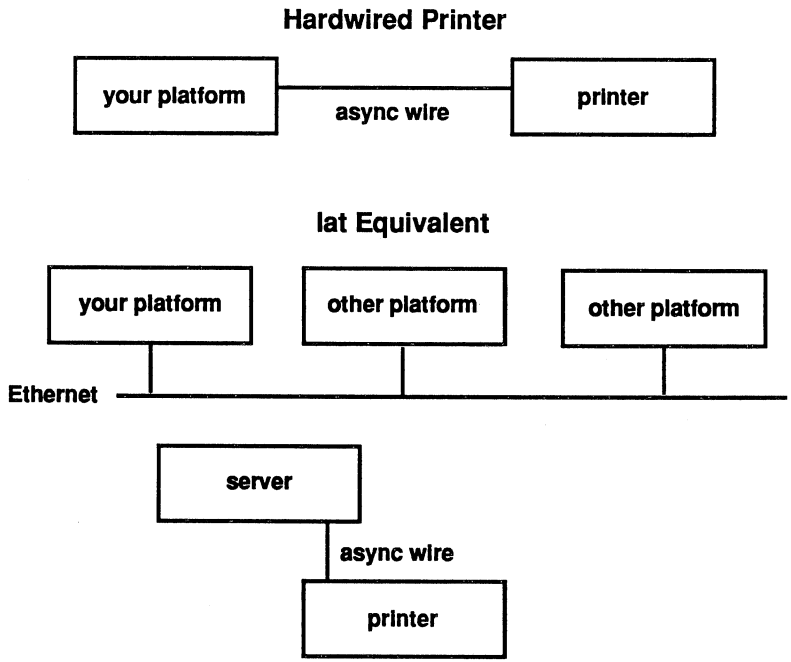
Once the port number and server name have been determined, you need only decide on a **/dev/filename** that you wish to use for identifying the **/dev/ pseudo terminal** returned by the **create port** command. In the following example, we are creating a printer port to which your print spooler can write.

```
create port D200 PORT_1 as latprint
Port created on '/dev/ttyp2' as '/dev/latprint'
```

You may direct printer output to the **latprint** device. You can also use the **/dev/ttyp2** device as well in your **print** command, although this is not recommended.

lat and Print Spoolers

The print spooler on your system may now direct output to the **latprint** device or **ttyp** device, as if the printer were directly connected to your system with an async wire.



LAT Host Services allow the printer attached to the terminal server to be used as needed by the print spooler on your system. **lat** makes the connection to the terminal server port and passes the print spooler output directly to it.

In the event that the printer is busy, the terminal server will queue your **lat**'s request until the printer is available. As the printer is made available, the print spooler output will complete. If the terminal server goes off line, an error will be returned.

When **lat** is running in the foreground, and the printer is outputting, the following error appears:

```
Virtual Circuit Timeout
Remote server address:  AA-BB-CC-DD-EE-FF
```

If the printer is not printing, no error is generated.

When the printer is down while trying to connect, the following error appears:

```
Unable to connect application port, no response from server
Dead ports device: xxx
Will try connecting again in 15 minutes
```

Creating the Printer Port at lat Initialization

When **lat** starts, it looks for the file **/usr/etc/.latrc**. If this file is present, it executes the commands in the same way as if **latcp** were running. To create a printer port when **lat** is first started, enter the **create port** command. For example,

```
create port 580R43 PORT7 as neon
```

printcap File

The file **/etc/printcap** is used to define parameters attached to your system. There must be an entry in this file for each printer port you have defined. For example,

```
lp|neon:\
        :lp=/dev/neon:sd=/usr/spool:lf=/usr/adm/lpd-errs:\
        fs#023:pl#64:pw#80:
```

See the **printcap** manpage for more details.

5. lat Statistics

The proper syntax for gathering statistics is listed in table 5-1. These commands may be entered from the **lat** command line, **LAT>**, or the **latcp** prompt (**latcp** interface to the forked **lat** process).

Table 5-1. lat statistics gathering commands

Command	Description
Show Executor	Show Executor Database and related counter values
Show Known Circuit Counters	Show Circuit Counter values
Show Known Line Counters	Show Line Counter values
Show Known Ports	Show Known Application (printer ports)
Show Known Slot Counters	Show Slot Counter values
Zero Executor Counters	Zero the Executor Counter values
Zero Known Circuit Counters	Zero Circuit Counter values
Zero Known Line Counters	Zero Line Counter values
Zero Known Slot Counters	Zero Slot Counter values

An example of each statistic gathering command and its related output are shown. The minimum number of characters is shown in the example.

Show Known Circuit Counters

```
LATCP> s k c c
Server Name:              xxx
Server                    AA-BB-CC-DD-EE-FF
Zeroed on:                Fri Nov 17 15:45:20 1989
Time since zeroed:        10 minutes 25 seconds
Local Circuit ID:         256 (0x0100)
Remote Circuit ID:        257 (0x0101)
Number of Slots:          7


                          Transmitted       Received
                          -----------       --------
Messages:                 70                10
Bytes:                    1200              128
Retransmit Messages:      10                0
```

```
Max Datalink Size:              1500 Bytes
Keep Alive Timer:               15 seconds
Next Message #:                 72
Last Received Message #:        10
Received Ack #:                 70
Data Ready:                     True

Total of 1 circuit
```

Show Known Line Counters

```
LATCP> s k l
Line Counters for LAN Unit #0
Zeroed on Fri Nov  3  00:09:09  1989
Time since zeroed:  20 minutes 59 seconds

                            Transmitted       Received
                            -----------       --------
Messages:                   0                 0
Bytes                       0                 0
Multicast Messages:         21                0
Multicast Bytes:            2310              0

No System Buffer:  0
Send Failure:      0
```

Show Known Ports

```
LAT> s k p
Server Ethernet Address: 00-00-00-00-00-00
Port       ID       0x1000F27C
Host       ID       0x0000
Server     ID       0x0000
Device              '/dev/ttyq5' as '/dev/lat1'
State               Wait_IO
status              0x0000
slotdb              0x1000F2F0
pty_fd              6
srvr                'VISTA'
port                'PORT_1'
msg_tmr             0 [0]
retransmits         0
Groups:
      0
1 application port
```

Show Known Slot Counters

```
LAT> s k s c
Slot Counters

Server Name:       VISTA
Server Address:    00-00-B5-00-02-B8
Slot:              0x10000EC5C
Local Slot ID      1 (0x0001)
Remote Slot ID:    1 (0x0001)
Process PID:       4094
Device:            '/dev/ttyq4'
      Internal fd:     5
Zeroed on:         Thu Nov 2 09:24:06 1989
Time since zeroed: 35 seconds

                          Transmitted       Received
                          -----------       --------
Messages:                 41                36
Bytes                     744               221
Data_A Messages:          40                34
Data_A Bytes:             561               17
Max Data Size #1:               255 bytes
Max Data Size #2:               255 bytes
Local Credits to Send:          0
Remote Credits Received:        2
Send Queue Length:              0 mbufs
Data Bytes Pending:             0 bytes
Output Building Length:         0 bytes
Output Off:                     False
Application Port:               0x00000000
Total of 1 slot
```

Show Executor

```
LAT> s e
LAT Executor Database:

Current Rating:          6
Current Time:            Fri Nov 3 00:34:51 1989
Time Zeroed:             Fri Nov 3 00:09:08 1989
Time since zeroed:       1543 seconds (25 minutes 43 seconds)
```

```
Counters
================
Bad Node                0 messages
Bad Response            0 messages
Bad Status              0 messages

Timers
================
Circuit                 80 ms
Hello                   60 seconds ( 1 minute )
Request                 10 seconds
Wait                    65 seconds ( 1 minute 5 seconds )

Configured Limits
================
Frame Size          1500 bytes
Slot Size           255 bytes
Max Sessions        16
Max Active Sessions 0
Active Sessions     0
Retransmit Limit    9
Low Water Mark      342 bytes (3 buffers @ 114 bytes each)
High Water Mark     798 bytes (7 buffers @ 114 bytes each)
Groups:
      0
```

lat Statistics Table

Table 5-2. lat statistics table

Command	Counter Name and Description
Show Known Line Counters	**Messages**: Total # of LAT messages. **Bytes**: Total # of bytes in the LAT messages. **Multicast Messages**: Of the messages, those that were multicast. **Multicast Bytes**: Of the bytes, those that were multicast. **No System Buffer**: # of times that a message was received and there was no buffer to receive it in. **Send Failure**: # of times that a transmit failed.

Table 5-2. lat statistics table (con't.)

Command	Counter Name and Description
Show Known Circuit Counters	**Server Name**: The name of the server which is connected to this circuit. **Server Address**: Ethernet address of Server. **Zeroed on**: Date/time that counters were zeroed. **Time since zeroed**: How long since the counters were zeroed. **Local Circuit ID**: Unique identifier for circuit, assigned by local node. **Remote Circuit ID**: Unique identifier for circuit, assigned by remote node. **Number of slots**: # of slots using this circuit. **Messages**: # of circuit messages. **Bytes**: # of circuit message bytes. **Retransmit Msgs**: # of circuit messages retransmitted. **Max Datalink Size**: # of largest transmit size allowed. **Keep Alive Timer**: 3 * server keep-alive-timer (to know when server dies). **Next Msg #**: Next circuit message number to transmit [0..255] **Last Rcvd Msg #**: LAT circuit message number received from remote [0..255] **Rcv'd ACK #**: Last ACK received **Data Ready**: true: data is waiting to be sent to server. false: no data waiting for server.

Table 5-2. lat statistics table (con't.)

Command	Counter Name and Description
Show Known Slot Counters	**Server Name**: The name of the server which is connected to this circuit. **Server Address**: Ethernet address of server. **Slot**: Internal use [address of slot db]. **Local Slot ID**: Slot identifier assigned by local node. **Remote Slot ID**: Slot identifier assigned by remote node. **Process ID**: Process ID of child login shell (not always accurate). **Device**: Login/application port device **Internal fd**: Internal use (file descriptor for the pty). **Zeroed on**: Date/Time that counters were zeroed. **Time since zeroed**: How long since the counters were zeroed. **Messages**: # slot messages. **Bytes**: # slot bytes. **Data_A Msgs**: # slot messages which were Data A slots. **Data_B Msgs**: # slot messages which were Data B slots. **Max Data Size #1**: Internal Use. **Max Data Size #2**: Internal Use. **Local Credits**: # of credits to give to remote node. **Remote Credits**: # of credits given to us from remote node. **Send Q Length**: # of buffers containing data waiting to be sent. **Data Bytes Pending**: # of bytes in Send Q. **Output Bldg Len**: # of bytes not yet dedicated to Send Q. **Output Off**: true: not accepting new output from login/application port. **Application Port**: Internal use (address of application port database).

Table 5-2. lat statistics table (con't.)

Command	Counter Name and Description
Show Known Ports	**Port ID**: Internal use (address of application port database). **Host ID**: Unique ID assigned to pending/ active port by local node. **Server ID**: Unique ID assigned to pending port by remote node. **Device**: Actual and logical device names for this port. **State**: State of port: Wait_io — Waiting for output from application before connecting. Search — Searching for remote node. Request — Requesting connection establishment. Pend — Connection request is pending. Run — Connection is active. Unused — Port is not in use. **Status**: Last received status from remote about pending request. **slotdb**: Internal use [address of slot database]. **pty_fd**: Internal use (file descriptor for pty). **srvr**: Name of remote node. **Port**: Port name at remote node. **msg_timr**: # of seconds left on timer and timer state (state is in []). **States**: 0 - stopped; 1 - active; 2 - expired **Retransmits**: # of times current message has been retransmitted. **Groups**: List of groups that this port is a member of.

Zeroing Counters

You may want to zero counters to get a clear picture of how the various counters may be growing or not growing in size. The command syntax and example commands are:

Zero Known Line Counters

```
LATCP> z k l c
Line counters zeroed
```

Zero Known Circuit Counters

```
LATCP> z k c c
Circuit Counters Zeroed
```

Zero Executor Counters

```
LATCP> z e c
Executor Counters Zeroed
```

Zero Known Slot Counters

```
LATCP> z k s c
Slot Counters Zeroed
```

Syntax

If you receive the following message try reentering the command or use **help** to get syntax.

```
LATCP> s l c
Unknown CONTROL command

LATCP> s c c
Unknown CONTROL command

LATCP> s pc
Unknown CONTROL command
```

6. Troubleshooting lat

This section contains information on two commands which assist in defining a **lat** problem. The commands are **scopefile** and **set debug**.

You may be asked to enter the **scopefile** or **set debug** command for later analysis by a support engineer. The explanation of the syntax of each command follows.

scopefile

The **scopefile** command allows you to direct **lat** to write every packet of transmit or receive to a specified disk file. To direct packets to **/build/usr/lat/lat_scope.trc** you enter:

```
LATCP> scopefile /build/usr/lat/lat_scope.trc
SCOPING is ENABLED to '/build/usr/lat/lat_scope.trc'
```

To stop the flow of packets to this file, you enter:

```
LATCP> scopefile
SCOPE file '/build/usr/lat/lat_scope.trc' closed
```

set debug

A support engineer may direct you to set a certain level of debug output. In order to do this, you would enter:

```
LATCP> set debug 2
Setting DEBUG to 2.
```

The **if** statements which don't compare with any number, will cause output to be generated for any level of debug.

```
if (KIDEBUG) then printf ("Received CONTROL command '%s'\n",
   msg);
if (KIDEBUG >=1) then printf ("Killing pid %d, slot is gone\n",
   slotdb->pid);
if (KIDEBUG >=6) then printf ("process_ptv_read: %d bytes\n",
   by tes);
if (KIDEBUG >=7) then bdump (stdout, bp, bytes);
if (KIDEBUG >=1) then printf ("process_pty_read:
   Mode=0x%02X\n", control_code);
if (KIDEBUG >=1) then printf ("pty read returned %d\n", bytes);
if (KIDEBUG >=7) then printf ("processing from lat_slave
   socket\n");
if (KIDEBUG > 0) then fflush (stdout);
if (KIDEBUG >=9) then printf ("setting lat_master:
   close(lat_master)\n");
```

```
if (KIDEBUG) then printf ("Setting Group %d.\n", j);
if (KIDEBUG >=1) then perror ("find_pty: calling k_getapty");
if (KIDEBUG >=1) then perror ("repoen_pty: finding pty");
if (KIDEBUG >=1) then printf ("process_pty_read: status=0x%-
   8X\n", status.all);
endif
if (KIDEBUG >=1) then
      {
      sprintf (errtxt, "===> errno: %d.\r\n", errno);
      slot_bdata_a_str (slotdb. errtxt);
      }
end_if
if (KIDEBUG >=1) then
      printf ("process_sock: status=0x%08X\n", status.all);
end_if
if (KIDEBUG >=3) then
      printf ("READ Event for ec_num %d, slotdb=0x%X,
            pty_fd=%d\n. ec_num, slotdb, slotdb-->pty_fd;
end_if
if (KIDEBUG >=1) then
      {
      if (not found) then
      printf ("Deaths were registered but none were
            found!\n");
      elseif (found not_equals deaths) then
            printf "(Insufficient dead pty's were found\n");
      end_if
      }
end_if
if (KIDEBUG >=3) then
      printf ("READ Event for pty_fd= %d\n", slotdb-->pty_fd;
end_if
if (KIDEBUG >=1) then
      {
      printf ("\nPID %d has died\n", pid);
#ifdef BSD
      printf {"\
      w_Retcode= %d\n\
      w_Coredumb= %d\n\
      w_Termsig= %d\n\
      ", status.w_retcode, status.w_coredumb,
            status.w_termsig);
#endif BSD
      }
end_if
```

```
if ((not found) && (KIDEBUG >=2)) then printf ("\ sig_child:
            WARNING, Unable to locate slotdb entry for pid
            %d\n", pid);
if (KIDEBUG >=1) then
      {
      printf ("Device '%s' is pty_fd %d\n", slotdb-->ttyname,
            pty_fd);
      }
end_if
if (KIDEBUG >=2) then
      printf ("app_port: Device '%s' is pty_fd=%d\n", slotdb-
            ->ttyname, slotdg-->pty_fd);
end_if
```

A. Glossary

advertising. The process that allows users to identify names and characteristics of the resources to be used. As applied in this document, the term "advertising process" refers to a certain message-exchange mechanism provided by the LAT architecture. Each node, whether it is a slave, a master or both, can advertise services.

application terminal. A device that is under control of the application process running within a slave environment. In some cases an application device may not have a keyboard or even be a "terminal" at all. It may be a line printer, a video monitor, a display window etc.

broadcast. As applied to data links, broadcast capability refers to the ability of any one port to address all other ports simultaneously with a single datagram.

datagram. An atomic unit of information exchanged by local area networks. In the Ethernet implementation, datagrams are required to have a constant format consisting of: destination port address, source port address, protocol type, data and an error detection code. Datagrams may get corrupted on the Ethernet, and are therefore not always delivered to the destination address.

flow control. A set of rules applied to processes which prevents a transmitting process from sending data to a receiving process that is not prepared to buffer the transmitted data.

interactive terminal. A device that is under the control of the terminal user connected to a node running in master mode.

local area (network). The topology defined by the set of logically equivalent processors directly attached to a shared interconnect.

master. An addressable process that provides communication attachment points for virtual circuits. The master initiates and controls activity over virtual circuits. The state-table of a master process is defined in the LAT architecture document.

message. A datagram under virtual circuit error control.

multicast. As applied to data links, multicast capability refers to the ability of any one port to address a subset of all other ports simultaneously with a single datagram.

name. A string of ASCII characters meaningful in the context of the client using it. Names are used to provide identification of entities within the LAT architecture that can and need to be identified. Characters within an ASCII string representing the name are constrained as described in the section of the LAT architecture entitled "Specification of Names."

node. The environment on the end of the virtual circuit that provides functioning of a master and slave process. A node can operate as slave, master or both simultaneously. In this document the expression *slave (master) node* really means *a node operating in slave (master) mode.* Each node is uniquely identified by name.

object. A provider of the resources; a passive responder to requests for establishing connections. An object does not initiate connections. An object can be a master as well as a slave.

port. An access point that a node presents to users. Each port serves as a communication path between a user and a resource. Ports can be named.

resource. An entity or set of entities known to perform a certain set of functions that can be identified, named and accessed within LAT.

service. The descriptive name of the resources. The name is used by users to identify a resource and is used by LAT to establish an access path to the resource.

service class. A 1-byte value in the range 0-255.

value 0 - reserved
value 1 - reserved for interactive and application terminals (serial byte stream processing)
values in the range 2 to 127 reserved for DEC use
values in the range 128-255 reserved for customers

session. A transient association that allows a subject to exchange data reliably with an object by utilizing an underlying shared virtual circuit.

session (connection). A transient association which allows a terminal server to exchange data reliably with a single host service utilizing and underlying shared virtual circuit.

slave. An addressable process that provides the passive side of the communication attachment point for a virtual circuit. The slave responds to the master's request. The state-table of a slave process is defined in the LAT architecture document.

slot. A segment of a message used to communicate data between a terminal on a terminal server and a host service. Messages may have zero or more slots.

subject. A consumer of the resources, an active initiator of a connection. A subject can initiate and support relations only with objects, not with other subjects. A subject can be a master as well as a slave.

terminal server. A dedicated function system (processor, controller) providing attachment points for terminals in the local area via a responsive virtual circuit service spanning the shared interconnect.

virtual circuit. A communication path between a master and a slave. A virtual circuit is a bidirectional, sequential, timely, and error-free logical stream of data. On Ethernet, a virtual circuit service is a value added service since the Ethernet data link provides a datagram service.

B. Installation

Before Installation

Before you install LAT Host Services, you must have the following information:

- The *Factory Ethernet Address* of your ESV Workstation. Enter the following command to get this address:

  /usr/etc/factaddr

  Factory Ethernet Address __ - __ - __ - __ - __ - __

- The LAT Host Services *Software Key Number* (18 digits). Call Ki Research to at **1-800-544-8352** to obtain this number. You will be asked for the *System Type* (ESV Workstation) and the *Factory Ethernet Address*.

  Software Key Number_____

General Notes

- You must be logged on the system as **root**.

Loading the LAT Host Services Software

Insert the LAT Host Services tape into the tape drive and enter the following command:

/usr/pkg/bin/inst

Following is an example which shows the loading of the LAT Host Services software.

```
MIPS software package installation

Install package relative to where [/]? <RETURN>
Please mount the (first, if multiple tapes) distribution
   tape, then press return...<RETURN>

Rewinding the tape... Verifying tape id... ok

Extracting packaging information tree... eslat1.0

========== selecting subpackages ==========

Install subpackage eslat (y n) [n]? y
Selected subpackages:
   eslat
Is this what you want (y n) [y]? <RETURN>

========== setting system clock/calendar ==========
```

```
The current value of the clock is: Tue Feb 27 15:52:54 MST
   1990
Is the clock correct (y n) [y]? <RETURN>

========== verifying single-user mode ==========

This system is not presently in a single-user run level.
   Installation of a package can fail if performed at this
   run level. We recommend that the system be brought to a
   single user run level (using "init S") prior to
   performing the installation.

Are you absolutely sure you wish to continue (y n) [n]? y

========== preserving local files ==========

No preserve list or findmods list for eslat- preserve not
   executed.

========== verifying disk space ==========

The system will now be checked to verify that there is
   enough disk space with the current configuration to
   successfully install the package (and any selected
   optional subpackages). For large packages (especially
   operating system packages), this can be time
   consuming...

There is enough space.

========== stripping old links ==========

Stripping links for subpackage eslat...

======= extracting files from subpackage archives =======

Rewinding the tape...
Verifying tape id... ok
Forward spacing the tape...

Loading subpackage: eslat...
Forward spacing the tape...
Rewinding the tape...

========== running comply ==========

running first comply pass...
running second comply pass...
There were no comply messages from the second pass.

========== cleaning up old versions ==========

An attempt will now be made to clean up any files left over
   from previous versions of the software which has just
   been installed.

Searching for old versions to remove...

========== restoring preserved user files ==========
```

```
No preserve list or findmods list for eslat- no files
   restored.

========== cleaning up ==========

Remove install tools (y n) [y]? <RETURN>

========== installation complete ==========
```

Installing the Software Key

Using any editor, create the file **/etc/kl_pwd**. Enter the Software Key Number on one line of this file, with no spaces before, or between, any of the 18 digits.

If you have previously installed the ES/Dnet software, this file will already exist. In this case, enter the ES/Dnet key after the existing key.

Installing the LAT Host Services Software

Run the installation script, which is located in the following file: **/usr/etc/eslat/bin/install.lat.**

Following is an example which shows the installation of the LAT Host Services software.

```
Ki LAT Installation procedure

ln -s /usr/etc/eslat/bin/lat /usr/etc/lat
ln -s /usr/etc/eslat/bin/latcp /usr/etc/latcp
ln -s /usr/etc/eslat/bin/uninstall.lat /usr/etc/
   uninstall.lat
```

If you want to know about options that can be used when starting LAT Host services, you can use the **-help** option (*i.e.*, **lat -help**).

```
Would you like to see this help display now? [Y/N]
n

Installation complete
```

To start LAT Host Services, enter the following command:

```
/etc/init.d/lat start
```