IBM Communications Server for Linux

Diagnostics Guide

Version 6.2.3

SC31-6779-02

IBM Communications Server for Linux

Diagnostics Guide

Version 6.2.3

SC31-6779-02

Note:
Before using this information and the product it supports, be sure to read the general information under Appendix E, "Notices," on page 73.

Fourth Edition (December 2007)

This edition applies to IBM IBM Communications Server for Linux, Version 6.2.3, and to all subsequent releases and modifications until otherwise indicated in new editions or technical newsletters.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

IBM welcomes your comments. You may send your comments to the following address:

International Business Machines Corporation
Attn: Communications Server for Linux Information Development
Department AKCA, Building 501
P.O. Box 12195, 3039 Cornwallis Road
Research Triangle Park, North Carolina 27709-2195

You can send us comments electronically by using one of the following methods:

- Fax (USA and Canada):

  1-919-254-4028

  Send the fax to "Attn: Communications Server for Linux Information Development."
- Internet email:

  comsvrcf@us.ibm.com

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Tables

Figures

About This Book

This book describes and offers solutions to the most common problems you might encounter while using IBM® Communications Server for Linux®. This book also introduces the diagnostic tools available to you, and explains how to collect diagnostic data for support personnel.

IBM Communications Server for Linux is an IBM software product that enables a computer running Linux to exchange information with other nodes on an SNA network.

There are two different installation variants of IBM Communications Server for Linux, depending on the hardware on which it operates:

Communications Server for Linux
Communications Server for Linux, program product number 5724–i33, operates on the following:

- 32–bit Intel® workstations running Linux (i686)
- 64–bit AMD64/Intel EM64T workstations running Linux (x86_64)
- IBM pSeries® computers running Linux (ppc64)

Communications Server for Linux on System z™
Communications Server for Linux on System z, program product number 5724–i34, operates on System z mainframes running Linux for System z (s390 or s390x).

In this book, the name Communications Server for Linux is used to indicate either of these two variants, and the term "Communications Server for Linux computer" is used to indicate any type of computer running Communications Server for Linux, except where differences are described explicitly.

This book applies to Version 6.2.3 of Communications Server for Linux.

Who Should Use This Book

This book is intended for System Administrators and application programmers who use Communications Server for Linux:

System Administrators
System Administrators install Communications Server for Linux, configure the system for network connection, and maintain the system. They should be familiar with the Communications Server for Linux system and with the hardware on which Communications Server for Linux runs. They must also be knowledgeable about the network to which the system is connected and understand SNA concepts.

Application programmers
Application programmers design and code transaction and application programs that use the Communications Server for Linux programming interfaces to send and receive data over an SNA network. They should be thoroughly familiar with SNA, the remote program with which the transaction or application program communicates, and the Communications Server for Linux system programming and operating environments.

More detailed information about writing application programs is provided in the manual for each API.

How to Use This Book

This section explains how information is organized and presented in this book.

Organization of This Book

This book is organized as follows:

- Chapter 1, "Overview of Problem Solving," on page 1, describes the diagnostic tools that are available to you, why they are important, and when to use them.
- Chapter 2, "Resolving Common Problems," on page 9, identifies problems you are most likely to encounter and provides step-by-step guidance on solving the problems.
- Chapter 3, "Using Logging and Tracing," on page 27, provides task-oriented descriptions of logging and tracing to enable you to gather information from the system. This chapter also includes detailed tracing procedures for the most commonly used traces.
- Appendix A, "Using snafilter and snatrcfmt," on page 53, describes how to use the **snafilter** and **snatrcfmt** utilities to format binary trace output.
- Appendix B, "Using getsense," on page 67, describes how to use the **getsense** utility to display sense codes online.
- Appendix C, "Using snagetpd," on page 69, describes the diagnostic collection utility (**snagetpd**) and how to use it to gather diagnostic information for support personnel.
- Appendix D, "Windows Clients," on page 71, contains Windows-specific information for Communications Server for Linux users.

Typographic Conventions

Table 1, shows the typographic styles used in this document.

Table 1. Typographic Conventions

Special Element	Sample of Typography
Document title	*Communications Server for Linux Administration Guide*
File or path name	**/var/opt/ibm/sna/sna.err**
Program or application	**vi**
Command or Linux utility	**define_default_pu**
General reference to all commands of a particular type	**define_*** (indicates all of the define administration commands)
Option or flag	**ALL**
Parameter or Motif field	*log_file_type*
Literal value or selection that the user can enter (including default values)	`USER,NODE`
Constant or signal	`ERROR`
Return value	`Audit`
Variable representing a supplied value	*server name*
Environment variable	`$DISPLAY`
Programming verb	REGISTER_NMVT_APPLICATION
User input	**xsnaadmin**
Computer output	**+RSP**
Function, call, or entry point	`Set Session Parameters`

Table 1. Typographic Conventions (continued)

Special Element	Sample of Typography
Motif button	**Status**
Motif menu	**Services**
Motif menu item	**Configure node parameters**
Keyboard keys	**Enter**
Hexadecimal value	`0x0a`

Related Publications

For information about SNA, APPN®, or LU 6.2 architecture, refer to the following IBM documents:

- *IBM Systems Network Architecture:*
  - *LU 6.2 Reference: Peer Protocols*, SC31-6808
  - *APPN Architecture Reference*, SC30-3422
  - *Management Services*, SC30-3346
  - *Formats*, GA27-3136
  - *Technical Overview*, GC30-3073

Chapter 1. Overview of Problem Solving

Communications Server for Linux is a complex software product. You may therefore occasionally encounter problems when you are running Communications Server for Linux, either with Communications Server for Linux itself, or with other system components.

This manual describes some of the more common types of problems that you might encounter, provides guidance for investigating them, and describes how to gather further diagnostic information. The manual is structured in the following way:

- This chapter introduces the types of diagnostic information that are available to you, and describes how to use each type.
- Chapter 2, "Resolving Common Problems," on page 9, describes the basic checks that you should always perform, and provides step-by-step guidance on how to investigate a specific problem further. Read this chapter if you have encountered a problem and need to know how to investigate it.
- Chapter 3, "Using Logging and Tracing," on page 27, describes how to use the Communications Server for Linux tracing and logging facilities to gather further diagnostics information. Read this chapter if you need guidance on how to gather a particular type of trace or how to gather specific kinds of logs.

Types of Diagnostic Information

This section describes the range of diagnostic information that is available to you to resolve Communications Server for Linux system problems, and how to use each type.

Communications Server for Linux diagnostic information can be categorized in the following way:

- "Notification" information is always available to you; it cannot be turned on or off. This type of information indicates that an error has occurred and should be investigated. It includes error messages, error logs and alerts.
- "Diagnostic" information can be controlled, and should be used to gather further information on specific problems. This type of information includes exception logs, audit logs, and tracing data.

Communications Server for Linux also provides online help information that can be useful in preventing or resolving problems.

When Communications Server for Linux is running, you will generally use all these types of information, at different times, to resolve any problems that you may encounter.

For example, when an APPC application program is running, different Communications Server for Linux events such as starting or stopping a session may occur. Each event is made up of a number of smaller events. So a session event can include internal events such as connecting to an LU, starting the session, security checking, and link initialization. You can configure Communications Server for Linux to log each of these smaller normal events to a file called an audit log file, if you wish to record them all.

Types of Diagnostic Information

When a program fails, Communications Server for Linux provides several diagnostic resources, each of which gives you different types and levels of information about the events that have occurred. This information can be displayed on your screen or logged to a file called an error log file. Further event information for a particular area of Communications Server for Linux can also be generated and collected in a file when you activate traces for a specific feature of Communications Server for Linux.

This section describes all these types of information and how you might use them.

Information from Program Error Messages
These messages are displayed by the system whenever a serious system problem is encountered.

Information from Communications Server for Linux Log Messages
Log messages contain information about program events. There are three different types of events about which information is logged: problem, exception, and audit. Each is captured to a log file that you can access to obtain the information needed to resolve a problem.

Communications Server for Linux also maintains a separate log file that records information about your usage of SNA resources on the local node. See "Usage Logging" on page 35 for more information.

Information from Communications Server for Linux Tracing
Tracing is a means of tracking the events occurring across a particular boundary of Communications Server for Linux while Communications Server for Linux is running. Communications Server for Linux offers a wide variety of tracing options that can be activated for diagnostic purposes.

Information from Alerts
Standard SNA alerts are generated and transmitted to the host. They can be viewed on the host with NetView®.

The program error messages and log messages usually specify the nature of the problem, its cause, and the recommended action, which is often enough to help you resolve a problem. The amount of information available to you depends on the nature of the problem and on how you set up logging. For example:

- Program error messages are automatically displayed; they cannot be disabled.
- Event logs indicating problems are also automatically generated and cannot be disabled, though logging of exception event logs can be disabled. Audit event logs are not logged unless you choose to log them.

Tracing should only be activated when you suspect a problem. You can then trace activity in the area that you are experiencing a problem with. Additionally, you can control the amount and format of the tracing information collected for you.

In addition to diagnostic information, Communications Server for Linux also has extensive online help information for the following:

- Motif administration program
- Command-line administration program

The rest of this chapter describes each kind of diagnostic information in more detail, and explains where to find Communications Server for Linux online help information.

For a list of the more common problems that are encountered by users and the steps to take to resolve them, see Chapter 2, "Resolving Common Problems," on page 9. For information about how to use logging and tracing, see Chapter 3, "Using Logging and Tracing," on page 27.

Program Error Messages

The most obvious indication of a problem is the display of a program error message. Communications Server for Linux generates program error messages automatically to report serious problems. Program error messages are reported to the screen or console. Each message describes the problem that the program encountered. You cannot disable these messages.

For example, if an administrator using the command-line administration program tried to start a node when there was no node configuration file on the server, a message would be displayed similar to the following:

```
$: snaadmin init_node
init_node command failed:
primary_rc = STATE_CHECK, secondary_rc = NODE_NOT_CONFIGURED
```

If an administrator using the Motif administration program tried to modify parameters in the Node Parameters window while the node was running, a pop-up message would be displayed similar to the following:

```
You cannot modify the node's parameters while it is not inactive.
```

Often a program error message is accompanied by log messages that provide additional information. For information about the error log file, see "Types of Log Information."

Log Messages

When a program is executing, different events such as starting or ending a session occur. Communications Server for Linux records log messages for these and other events in log files to provide you with specific information about the internal activities of Communications Server for Linux. Other internal activities of Communications Server for Linux that are logged include port initialization, security checks, and network link station failures.

Communications Server for Linux logs messages for normal events (such as starting the session) and for abnormal events (such as unexpected session termination and resource shortage). For each event, the messages describe what happened, when, and where. You can disable some types of logging (see "Determining Which Messages Are Logged" on page 29) and control the amount of detail recorded in the logs (see "Choosing the Format of Logs" on page 30). One log file can contain more than one type of message.

Types of Log Information

Communications Server for Linux categorizes events by severity and groups them into one of three types.

Problem

An abnormal system event that degrades system performance in a way that is easily perceived by a user (for example, abnormal termination of a session).

Communications Server for Linux always logs these events. You cannot disable logging of these events.

Exception

Exception events fall into two categories:

- Abnormal system events that degrade system performance but are not immediately perceived by a user (for example, a resource shortage).
- Events that do not degrade system performance but may indicate the cause of later exceptions or problems. An example is receiving an unexpected message from the remote system.

By default, Communications Server for Linux logs exception events.

To control logging exception events using the Motif administration program, see "Controlling Logging Using the Motif Administration Program" on page 27.

You can also control logging of these events by using the **snaadmin set_global_log_type** command to establish global default settings for all servers, or the **snaadmin set_log_type** command to override the defaults for a particular server.

For more information about controlling logging using administration commands, refer to the *Communications Server for Linux Administration Command Reference*.

Audit A normal system event (for example, starting a session). By default, Communications Server for Linux does not log these events.

To control logging audit events using the Motif administration program, see "Controlling Logging Using the Motif Administration Program" on page 27.

You can also control logging of these events by using the **snaadmin set_global_log_type** command or the **snaadmin set_log_type** command.

Some problem and exception messages, which may require corrective action, are displayed on the Linux system console as well as being written to the log file. You can suppress these console messages by using the **-s** option when starting the Communications Server for Linux software (refer to the *Communications Server for Linux Administration Guide* for more information). Messages are then written only to the log file.

Using Log Information

In general, you should review the error log file first to help resolve any problem you encounter when using Communications Server for Linux.

The log files contain a *Message action* field, which describes any recommended action as a result of the message. In some cases, no action is required. For example, an exception message may not indicate an error but may provide background information that helps to identify the cause of a later problem message.

Common recommended actions include the following:

- Check the local Communications Server for Linux configuration and add, modify, or activate resources. For more information, refer to the*Communications Server for Linux Administration Command Reference*.
- Check the Linux computer's resources (such as memory, hard disk space, or adapter cards). For more information, refer to your Linux operating system documentation.

- Contact support personnel for the system with which Communications Server for Linux is communicating to resolve configuration mismatches. For more information, see "Types of Support Personnel" on page 22.
- Contact the developer of an application that uses the Communications Server for Linux APIs if the application is making API calls that are not valid.
- Report the error condition to your support personnel if the *Cause type* field indicates an internal error in the Communications Server for Linux software.
- Checking for logging information on the local system and on the remote server.

If the logs contain less information than you expect, you may have succinct logging enabled (succinct and verbose are the two modes available). To recover the missing information, use the **snahelp** utility (see "Using snahelp for Succinct Logging Messages" on page 33). For information about enabling verbose logging, see "Choosing the Format of Logs" on page 30.

Introducing Tracing

Communications Server for Linux provides trace facilities to enable you to capture and obtain information about the internal activities of Communications Server for Linux while it is running. This information can be helpful for diagnosing specific problems. For example, if you suspect a DLC (data link control) problem, you can activate line tracing for the specific DLC.

You can control the type and amount of trace data to be collected by using the administration programs. Depending upon the type of tracing you perform, your output is generated in either ASCII or binary format. Use a text editor to view ASCII files. For binary data, use the **snatrcfmt** utility to convert binary files to text files. For more information, see "Formatting Internal Binary Trace Output into Text Files" on page 57.

The following section briefly describes the types of tracing and provides examples of when to use tracing. For detailed information about tracing, see "Tracing" on page 36.

Types of Tracing

Communications Server for Linux provides the following types of tracing:

Line tracing

Use line tracing to trace messages between the node and the remote system. You can control the amount of tracing by specifying the resource type (DLC, port, link station, or session). Line Tracing is also sometimes known as "DLC Tracing".

Line tracing output is typically used by an SNA administrator to solve end-user problems, including the inability to bring up a session or session failure. The format is a standard SNA trace. For more information about line tracing, see "Line Tracing" on page 37.

The output produced by line tracing is binary. You can select specific entries from a line trace file by using the **snafilter** utility, and you can format binary trace output into text files by using the **snatrcfmt** utility. For more information about these utilities, see Appendix A, "Using snafilter and snatrcfmt," on page 53.

API tracing

Use API tracing to locate communication problems that involve any of the following Communications Server for Linux APIs: Advanced Program-to-Program Communication (APPC), Common Programming

Interface for Communications (CPI-C), Conventional LU Application Programming Interface (LUA), Node Operator Facility (NOF), Management Services (MS), and Common Service Verbs (CSVs). API tracing traces all the parameters supplied to the API library or driver by an application and all the parameters returned by the API library.

API trace data is written to text files. For more information, see "API Tracing" on page 40.

Client/Server tracing

Use client/server tracing to trace messages flowing between the Communications Server for Linux server and a client, as well as between Communications Server for Linux servers in the same domain. Client/server tracing can be used, for example, to find out why a client is unable to connect to the server.

Client/server tracing is written to text files. For more information, see "Client/Server Tracing" on page 47.

TN server tracing

Use TN server tracing to record messages flowing between the Communications Server for Linux TN server and its TN3270 clients. Typically, the System Administrator initiates this trace to resolve a TN3270-related problem.

The TN server trace data is written to text files. For more information, see "TN Server Tracing" on page 49.

Internal tracing

Use internal tracing to trace messages that flow between internal components of the Communications Server for Linux node.

A large amount of trace output will usually be generated quickly. Once you have captured the information you need to solve your problem, stop tracing to prevent files from growing too large or from being overwritten. For more information about internal tracing, see "Internal Tracing" on page 50.

The output produced by internal tracing (which is typically used by support personnel) is binary. You can select specific entries from an internal trace file by using the **snafilter** utility, and you can format binary trace output into text files by using the **snatrcfmt** utility. For more information about these utilities, see Appendix A, "Using snafilter and snatrcfmt," on page 53.

Using Tracing

If you encounter a problem, use the Communications Server for Linux trace facilities to obtain more information about the messages that are flowing across specific interfaces. For example:

- If you have an application that uses the Communications Server for Linux APIs and the API return codes indicate a problem, use API tracing.
- If you are unable to connect to a remote system successfully, or if Communications Server for Linux produces exception and problem logs when you try to do so, use line tracing or client/server tracing.

Note: Do not run Communications Server for Linux tracing unless you have encountered a problem with your Communications Server for Linux system. The system runs more slowly when tracing is enabled.

For more information about using tracing, see "Tracing" on page 36.

Alerts

Alerts are SNA messages that are generated automatically. They are sent to and processed by different programs on the external network and are used to identify problems or impending problems. There can be alerts from connectivity components or alerts provided by an application program using the Management Services API.

Alerts are sent to the first active host link encountered or to the link station you specified using the **snaadmin define_default_pu** command. If the link station is inactive, alerts are stored on disk and transmitted when the link station is reactivated.

NetView, which typically runs on the host, is the most commonly used program for viewing and processing alerts.

Help Information

In addition to log and trace information, which describe Communications Server for Linux system activities that occur during a problem, Communications Server for Linux also includes standard online help information. You may find this information useful if you encounter problems while using a particular program or if you want more information about a certain topic or about using a command.

Communications Server for Linux provides the following online help information:

- Motif administration program online help
- Command-line administration program help
- Communications Server for Linux manuals in PDF format
- Man pages
- Usage strings

Motif Administration Program Online Help

The Communications Server for Linux Motif administration program, which is used to configure and manage Communications Server for Linux, provides online help. You can access this online help in two ways; each provides a different form of help information:

- From the main window you can access a broad range of help topics.
  1. To display the online help menu, click **Help** in the top right-hand corner of the administration window.
  2. Select **Contents** from the **Help** menu. The main online help dialog provides a list of topics.
- From individual dialogs you can access help on that particular dialog.
  1. To obtain more information about the dialog fields click on the **Help** button.

Command-Line Administration Help

You can access administration command-line help for a command through the command-line administration program. The help provides information about the full syntax of **snaadmin** commands, including their parameters and default values. For example, type the following command to obtain a description of the help that is available:

snaadmin -h

Communications Server for Linux manuals in PDF format

The manuals included on the installation media for this product are in Portable Document Format (PDF). PDF enables you to search, browse, or print the information easily, using hypertext links for related information. It also makes it easier to share the library across your site, because PDF viewers are available for many different platforms.

If you choose to install the PDF manuals when installing the product, they are installed in the directory **/opt/ibm/sna/docs**.

You can read the PDF manuals using any PDF viewer. For example, to view a PDF file on Linux, you can use **xpdf**:

xpdf *filename***.pdf**

Man Pages

Man pages contain information about command usage. You should access them if you need a quick overview of a topic or information about how to use a particular command.

To get a man page from the command line, type **man —M /opt/ibm/sna/man** *pagename*, where *pagename* represents the name of the page you want to see (generally the name of a Communications Server for Linux command). For example, to see the “sna” man page, type **man —M /opt/ibm/sna/man sna**.

In addition to displaying the requested information, man pages provide a list of related commands.

Usage Strings

You can access usage strings for Communications Server for Linux programs by invoking the programs using the **-h** parameter to obtain a brief syntax statement. For example, type the following:

sna -h

Chapter 2. Resolving Common Problems

This chapter identifies some of the more common problems that users may experience when running Communications Server for Linux. The steps suggested here should help you to resolve most of these problems.

This chapter describes the following:

- Basic checks that you should always perform whenever you experience a problem
- Specific problems and the steps that you should take to investigate them further
- Online support information for Communications Server for Linux
- The procedure for reporting a problem through your support channel if you are unable to resolve the problem yourself by following the steps described

Some of the checks that you will need to perform require you to check configuration information using either the Motif administration program or the command-line administration program. This manual describes what you need to check but you should refer to the following for more details on how to check it:

- If you are using the Motif administration program, refer to its online help for detailed information on particular tasks and fields
- If you are using the command–line administration program, refer to its online help for detailed information on the commands and their parameters, and also to the *Communications Server for Linux Administration Command Reference*

Basic Checks

You should perform the following basic checks, whenever you experience a problem. Perform each of the checks described in this section before moving on to "Resolving Specific Problems" on page 17.

What to Check First

Before suspecting a problem with SNA or Communications Server for Linux, take a few moments to complete the following checks:

1. Ensure that all communications cables are attached properly to your machine, switches, and hubs.
2. Check that all modems, switches, and hubs are enabled.
3. Check that the remote systems that you are trying to contact are active.
4. Check that your software and all optional software packages are installed correctly.
5. Check that all appropriate software fixes or patches have been applied.
6. Consult your system-specific installation documentation for information about known problems.

Check That the SNA Daemon Is Started

The SNA daemon programs must be started before you can use any Communications Server for Linux function, including the administration tools.

The system should start the SNA daemons automatically when Communications Server for Linux has been installed successfully, and whenever Linux restarts.

To start the SNA daemons (or to check whether they are running), issue the **sna start** command. This command starts the SNA daemons which include:

- `snadaemon`
- `snaerrlog`
- `snacfgdae`

The SNA daemons may fail to start if there is an error in the configuration, such as a reference to a missing configuration record. Other errors can occur if Communications Server for Linux has not been installed correctly. For example:

- There may be no domain configuration file (**sna_domn.cfg**).
- A required component such as LiS Streams has not been installed.
- Communications Server for Linux was not installed successfully.

Refer to *Communications Server for Linux Quick Beginnings* for more information, or refer to the **README** file on the Communications Server for Linux install image for help with any error messages that appear.

Check That the Local SNA Node Is Active

The local SNA node must be activated before Communications Server for Linux can become fully functional. To check whether the node is active, use the **snaadmin status_node** command. Refer to the *Communications Server for Linux Administration Guide* for more information about this command.

This command can report the following:

- The node is active.
- The node is inactive.

The Local SNA Node Is Active

If the **snaadmin status_node** command reports that the node is active then the node has been activated successfully.

The Local SNA Node Is Inactive

If the **snaadmin status_node** command reports that the node is inactive then you must activate the node using either the Motif administration program or the command-line administration program. Refer to *Communications Server for Linux Administration Guide* for information about starting the node.

If the node fails to start then check the error log file. The error log file might indicate that there is an error in the configuration file. In this case use the Motif administration program or the command-line administration program to correct the error.

For example, if the value of the *node_type* parameter of the **snaadmin define_node** command is changed from `NETWORK_NODE` to `END_NODE` without deleting the **topology.dat** file then an error is reported when the **snaadmin init_node** command is issued.

For more information about configuring the node, refer to *Communications Server for Linux Administration Guide*.

Check Communication with Other SNA Nodes

To check communication with other SNA nodes, check the following:

1. Check that links to other SNA nodes have been established.

- If you are using the Motif administration program:
  a. Check that the port that you are trying to use is marked as active.
  b. Check that the link station that you are trying to use is marked as active. The time at which the link station should be activated depends on the following configuration values:
     – Initially

       The link station is activated when the node is started.
     – Administrator

       The link station is activated by the administrator. (This may be a **snaadmin start_ls** command located in a startup shell script).
     – Incoming

       The link station is started from a remote node. Check that the remote node has attempted to start the link station.
     – On demand

       The link station is started only when required by an application or terminal emulator. (In this case it is still possible to override this in order to test the LS by starting the LS explicitly, as described below.)
- If you are using the command-line administration program display the status by entering the **snaadmin status_connectivity** command. Refer to the *Communications Server for Linux Administration Command Reference* for more information about issuing this command.

  The output from the **snaadmin status_connectivity** command shows the ports displayed on the Motif interface window grouped under "DLCs", each of which represents a physical communications adapter that supports one or more ports. The Motif administration program automatically defines and starts the DLC components when ports are defined and started. When using the command-line administration program, however, you must explicitly define and start the DLCs to which the ports belong.

2. If links to other SNA nodes have not been established, then start the DLC/port or link station. Refer to the *Communications Server for Linux Administration Guide* for more information about starting DLCs/ports and link stations.
3. If a DLC/port or LS is still not active, then refer to the sections below.

DLC/Port Fails to Start

If the DLC/port fails to start then you should check the error log file.

Also check the Communications Server for Linux error log file, which is normally **/var/opt/ibm/sna/sna.err** (unless you have used the administration tools to specify a different filename or directory).

If you are still unable to resolve the problem then the following checks will help you to locate the most common causes of the problem.

1. Check that the required adapter cards and associated device drivers have all been installed correctly. (Refer to the documentation that came with these products.)
2. Check that the configured adapter card number (the *adapter_number* parameter) matches that used by the hardware. For more information, refer to the section on defining port and DLC configuration parameters in the *Communications Server for Linux Administration Guide*.

   Check the DLC configuration in one of the following ways:
   - If you are using the Motif administration program, zoom on the DLC/port.

- If you are using the command-line administration program, issue the **snaadmin query_dlc** command to view the DLC configuration.

3. If your adapter card can support more than one port, check that the configured port number (the *port_number* parameter) matches that used by the hardware. For more information, refer to the section on defining Port and DLC configuration parameters in the *Communications Server for Linux Administration Guide*.

   Check the port configuration in one of the following ways:

   - If you are using the Motif administration program, then zoom on the DLC/port.
   - If you are using the command-line administration program, issue the **snaadmin query_port** command to view the port configuration.

4. If you have remote equipment, check that the appropriate cables are plugged in and that the equipment being connected is active.
5. If you are using SDLC switched lines, check that the modem is displaying data set ready (DSR) and carrier detected (CD). If these signals are inactive for long periods, there is probably a problem with either the modem or the line.
6. If you are using an MPC link, use the configuration and management tools provided with your Channel adapter card to check that it is active.
7. If you are using an Enterprise Extender (HPR/IP) link, check that the local IP address is specified correctly, both in the Communications Server for Linux port configuration and in the computer's IP configuration.

Link Station Fails to Start

If the link station fails to start then you should check the error log file.

Also check the Communications Server for Linux error log file, which is normally **/var/opt/ibm/sna/sna.err** (unless you have used the administration tools to specify a different filename or directory).

If you are still unable to resolve the problem then the following checks will help you to locate the most common causes of the problem.

1. Check that the remote system is active, as well as any intermediate routers or bridges.
2. Check the link station configuration in either of the following ways:

   - If you are using the Motif administration program, then zoom on the link station.
   - If you are using the command-line administration program then issue the **snaadmin query_ls** command to view the link station configuration.

   Check the following:

   a. Check that the node id (the *node_id* parameter) and control point name fields (the *adj_cp_name* parameter) (which are exchanged in XIDs) match between two systems that are trying to communicate.
   b. Check that the link role (primary/secondary/negotiable in the *ls_role* parameter, if applicable for the link type you are using) is compatible between the two systems that are trying to communicate. Any combination should work except for primary to primary and secondary to secondary.

3. If your link station still fails to start then check the sections below for your particular link type.

SDLC Link Station Fails to Start: If your SDLC link station fails to start, perform the checks listed in "Link Station Fails to Start" on page 12 before performing the checks listed below.

1. Check your modem configuration:
   a. Check that Communications Server for Linux is raising the data terminal ready (DTR) signal. (You may be able to see this on the modem status lights or display.) If Communications Server for Linux is not raising the data terminal ready signal, then check that the card and port number configured on the port have been set correctly.
   b. Check that the modem has been programmed for synchronous mode (because SDLC communication requires synchronous mode).
   c. Check that you have configured the dial string correctly.
2. Check the following configuration parameters:
   a. Check that the *NRZ/NRZI* parameter configured on the link station is set to the value used on the link.
   b. Check that the full/half duplex and constant carrier configuration settings are correct for the link. Most systems with modern modems are set up to be half duplex and constant carrier.
   c. Check that the link address is correct in the link station or port configuration. This is particularly important for multi-drop links.

   Check the link station configuration in one of the following ways:
   - If you are using the Motif administration program, then zoom on the link station.
   - If you are using the command-line administration program then issue the **snaadmin query_ls** command to view the link station configuration.

QLLC Link Station Fails to Start: If your QLLC link station fails to start, perform the checks listed in "Link Station Fails to Start" on page 12 before performing the checks listed below. Also check that the underlying X.25 software is active.

If you are unable to start a QLLC link station using a switched virtual circuit, then it is likely that the call request packet (sent out to establish a link station) carries incorrect parameters. You should therefore carry out the following checks:

1. Check the following configuration parameters:
   a. If you are setting up Communications Server for Linux to receive incoming calls then it is vital that the X.25 software is set up to route these calls to Communications Server for Linux. Check that the *incoming_address* parameter (if used) is configured correctly on the link station's port to match the local node's DTE address.

      Check the port configuration in one of the following ways:
      - If you are using the Motif administration program, then zoom on the port.
      - If you are using the command-line administration program then issue the **snaadmin query_port** command to view the port configuration.
   b. Check whether the remote system requires particular facilities or user data parameters. If it does, ensure that these are set appropriately in your Communications Server for Linux configuration of this link station.

      Check the Communications Server for Linux link station configuration in one of the following ways:
      - If you are using the Motif administration program, then zoom on the link station.

- If you are using the command-line administration program then issue the **snaadmin query_ls** command to view the link station configuration.

2. Check that the remote DTE address matches the address of the remote system.

Token Ring or Ethernet Link Station Fails to Start: If your Token Ring, or Ethernet link station failed to start, perform the checks listed in "Link Station Fails to Start" on page 12 before performing the checks listed below.

Check the following link station parameters:

1. Check that the remote MAC address, configured in the *mac_address* parameter, matches that of the remote system.
2. Check that the local and remote SAP configuration by doing the following:
   a. Check that the local SAP (configured in the link station's port *lsap_address* parameter) matches the remote system's remote SAP.
   b. Check that the remote SAP (configured in the link station's *lsap_address* parameter) matches the remote system's local SAP.
3. If the link station is an Ethernet link station then ensure that the LAN type (such as 802.3 or standard) matches that used by the remote system.

Check the link station configuration in one of the following ways:

- If you are using the Motif administration program, then zoom on the link station.
- If you are using the command-line administration program then issue the **snaadmin query_ls** command to view the link station configuration.

MPC Link Station Fails to Start: If your MPC link station fails to start, perform the checks listed in "Link Station Fails to Start" on page 12.

Enterprise Extender (HPR/IP) Link Station Fails to Start: If your Enterprise Extender (HPR/IP) link station fails to start, perform the checks listed in "Link Station Fails to Start" on page 12 before performing the checks listed below.

Check the following link station parameters:

1. Check that the IP address corresponding to the *remote_hostname* parameter is still valid. The hostname is resolved to an IP address at the time it is defined, or when the Communications Server for Linux software is stopped and restarted. If the remote computer is using DHCP, it is possible that its IP address may have changed; if this is the case, either stop and restart Communications Server for Linux or delete and redefine the link station to pick up the new IP address.
2. If you have more than one network interface card in your machine, check that the link station is configured on the correct port. You must be able to reach the IP address of the remote host from the local IP address corresponding to the port the LS is associated with.
3. Check the local and remote SAP configuration by doing the following:
   a. Check that the local SAP (configured in the port's *lsap_address* parameter) matches the remote system's remote SAP.
   b. Check that the remote SAP (configured in the link station's *lsap_address* parameter) matches the remote system's local SAP.
4. Check that the remote host supports the Enterprise Extender (HPR/IP) function.

Check the Communications Server for Linux link station configuration in one of the following ways:

- If you are using the Motif administration program, then zoom on the link station.
- If you are using the command-line administration program then issue the **snaadmin query_ls** command to view the link station configuration.

Note: The Motif administration program does not allow you to view the local and remote SAP addresses. To check these parameters, use the command-line administration program.

Check that the AIX or Linux Remote API Client Can See its Servers

First, refer to the chapter on managing clients in the*Communications Server for Linux Administration Guide*.

A client must be connected to a server before it can operate. Check that the client can see one or more servers by issuing a **snaadmin query_node_all** command from the client. (Refer to *Communications Server for Linux Administration Command Reference* for more information about this command.) Also check the file **server.current** that records the name of the server, if any, to which the client is currently connected. For a Remote API Client on Linux, this file is stored in **/etc/opt/ibm/sna**; for a Remote API Client on AIX®, this file is stored in **/etc/sna**.

The following sections describe the different responses to the **snaadmin query_node_all** command, and the actions that you should take.

List of Active Nodes Displayed

If the **snaadmin query_node_all** reports a list of all the active nodes (that is, servers), including the one named in the **server.current** file, then the client is able to see the servers and this is not the cause of the problem.

If one or more nodes is not in the list of active nodes, then ensure that each of these nodes is active. Refer to "Check That the Local SNA Node Is Active" on page 10 for more information.

If each node is active, but the list of active nodes displayed is still incomplete, then there may be a problem with the underlying TCP/IP network that Communications Server for Linux uses for its client/server communication.

To enable the client to continue to operate even when its server is inactive, you can configure backup servers or add more servers that the client is able to use by issuing the **snaadmin add_backup** command. Refer to the *Communications Server for Linux Administration Command Reference* for more information about this command.

No Active Nodes Displayed

If the **snaadmin query_node_all** does not report any nodes, or if the **server.current** file is empty (indicating that the client is not connected to a server), check the following:

1. Check that the domain name and the name of the server were both specified correctly when the client was installed by doing the following:
   - Issue the **snaadmin query_sna_net** command on the server
   - Check the **sna_clnt.net** file on the client

If the names do not match in both locations then modify the names in the **sna_clnt.net** file on the client so that they do.

2. Check whether the client has been configured to locate its server using broadcasts on a TCP/IP network that is not set up to route broadcast messages from the client to the server. If this is the case then specify the server name explicitly in the **sna_clnt.net** file.
3. Check whether there are any active servers by issuing **sna start**, followed by **snaadmin status_node** on the server.

If you have carried out all of these checks and fixed any problems, and the **snaadmin query_node_all** still does not report any nodes, then there may be a problem with the underlying TCP/IP network that Communications Server for Linux uses for its client/server communication. In this case you should consult your System Administrator.

Check that a Remote API Client on Windows Can See its Server

A Windows® client requires the services of a server before it can operate.

Check that the Windows client is started. Refer to the *Communications Server for Linux Quick Beginnings* for information about how to check this and how to start the Windows client.

If the Windows client is started but is still not working, check the following:

1. Ensure that you have installed and configured client support on the server by issuing **snaadmin query_sna_net**.
2. Check whether the Windows client is connected to the TCP/IP network and can see a server (use the **ping** command).
3. Ensure that the Windows client has been installed and configured correctly (refer to the *Communications Server for Linux Quick Beginnings* for information about installing Windows clients, and refer to the *Communications Server for Linux Administration Guide* for information about configuring Windows clients). In particular, if the client is running on Windows Vista, check that the firewall has been reconfigured to allow traffic on the appropriate TCP port; see the Windows client information in the **README** file on the installation CD for more details.
4. Ensure that the Windows client TP information has been configured correctly (refer to the *Communications Server for Linux Administration Guide* for information about configuring TP information on Windows clients).
5. Ensure that Communications Server for Linux has been started successfully on the Linux server.
6. Check the status of the Windows client by moving your mouse to the Communications Server for Linux icon on the toolbar (without clicking). One of the following tooltips, indicating the client status, will be displayed:
   - `Not started` indicates that you must start the client.
   - `Not connected` indicates that the client is running but could not connect to the network. You should investigate why the client was not able to connect to the network.
   - `idle` indicates that the client currently has no active sessions with the server. The client will reconnect automatically when you try to start a session, so no operator intervention is required.

- *servername* indicates that the client is active and has a server called *servername*.

Check System Configuration Information

Always check your system configuration before suspecting code problems. SNA configuration information is stored in the **/etc/opt/ibm/sna** subdirectory.

Configuration information is stored in the following text files:

- The **sna_node.cfg** file stores node configuration information
- The **sna_domn.cfg** file stores domain configuration information

You can modify the configuration information held in these files using one of the following:

- The Communications Server for Linux Motif administration program.
- The Communications Server for Linux command-line administration program.
- Client/server configuration is held in the **sna.net** file on the server, and can be viewed and modified using the Motif administration program or the command-line administration program.
- Linux client configuration information is held in the **sna_clnt.net** file.
- Windows client configuration information is held in the Windows Registry.

For more information about system configuration, refer to the *Communications Server for Linux Administration Guide*.

Resolving Specific Problems

If the problem persists after you have completed all the basic checks listed in "Basic Checks" on page 9 then this section provides further guidance on specific problems.

Continue your investigation by working through the sections that are most relevant to your problem.

Resolving Problems with Programs That Use Motif

If a Motif Communications Server for Linux program (such as the Motif administration program) fails to start then check the following:

1. Check that the Motif software is installed on your system. Refer to **README** file on the Communications Server for Linux install image for information about the required Motif version and how to install it.
2. Ensure that you are using a terminal with X-server support.
3. Check that address of your X-Server is configured on the X software on the Linux system on which you are running the Motif administration program. You can set this in either of the following ways:
   - Set the `DISPLAY` environment variable. For example, if you are using the Korn shell to connect to an X-Server whose TCP/IP name is "my_PC", type the following:

     export DISPLAY=my_PC:0
   - Specify the **-d** option when starting the Motif program. For example, if you are starting the Motif administration program to connect to an X-Server whose TCP/IP name is "my_PC", type the following:

xsnaadmin -d my_PC:0

4. Some X-Servers (such as servers running Linux) do not accept requests by default. If the Motif program reports that it is unable to connect, then you must configure it to accept requests. For example, if you are using X-Server running Linux, type the following:

   xhost +

Resolving APPC or CPI-C Application Problems

If you are having problems with an APPC or CPI-C application, including a Java™ CPI-C application, do the following:

1. Check that the application is correctly installed.
2. For a Java CPI-C application, check that the appropriate environment variables are set correctly, as follows.

   Before compiling and linking a Java CPI-C application, you need to specify the directory where Java classes are stored. To do this, set and export the environment variable `CLASSPATH` to **/opt/ibm/sna/java/cpic.jar**.

   Before running a Java CPI-C application, you need to specify the directory where libraries are stored, so that the application can find them at run time. You also need to set an additional environment variable to ensure that Java CPI-C works correctly with LiS Streams.

   To do this, set and export the environment variables as follows.

   For a 32–bit application:

   export LD_LIBRARY_PATH=/opt/ibm/sna/lib

   export LD_PRELOAD=/usr/lib/libpLiS.so

   For a 64–bit application:

export LD_LIBRARY_PATH=/opt/ibm/sna/lib64

export LD_PRELOAD=/usr/lib64/libpLiS.so
export PATH=/opt/ibm/java2-ppc64-50/jre/bin:/opt/ibm/java2-ppc64-50/bin:$PATH

   You may also need to set and export the `APPCTPN` to specify the local TP name for the application, as described in *Communications Server for Linux CPI-C Programmer's Guide*.

3. Check that the mode, LUs and link required are correctly configured.
4. If a problem occurs when the application originates the conversation do the following:
   a. Check that the link is active (this may happen on demand). Refer to "Check Communication with Other SNA Nodes" on page 10
   b. Check that the relevant session is started (this may happen on demand)
5. If a problem occurs when the application receives an incoming conversation request, do the following:
   a. Check that the link is active (this may happen on demand if the link is configured as incoming)
   b. Check that the relevant session is started (this may happen on demand)
   c. If the receiving application is dynamically loaded (invokable), also check that the TP configuration information has been configured correctly (refer to the *Communications Server for Linux Administration Guide* for more information about configuring TP information).

6. Verify that the session limit is sufficient for the number of applications required.
7. Check that you are not mixing single–session and parallel–session modes on an LU-LU pair.
8. The APPN architecture does not support independent LU 6.2 (parallel sessions) across a link station (typically SDLC) that is established without XID exchange (that is, with just SNRM and UA). Therefore, if the host does not send XIDs, independent LU 6.2 cannot be used.
9. Check the correct allocate type (for example, an immediate allocate needs a contention winner session).
10. For CPI-C applications, set the local LU name (using the **snaadmin define_cpic_side_info** command or `APPCLLU` environment variable) and TP name (`APPCTPN`), or use the default LU pool and TP name.

Resolving LUA Problems

For LUA, do the following to determine why the application is not working:

1. Check that the application is correctly installed and that you have permission to run it.
2. Use one of the following methods to ensure that the application is running:
   - If you are using the Motif administration program, check whether an application identifier or user name is displayed next to the LU.
   - If you are using the command-line administration program, issue the **snaadmin query_lu_0_to_3**, and check that the *appl_conn_active* parameter is set to `YES`.
   - You can also use the Linux **ps** utility. See "Other Information to Provide to Support Personnel" on page 23, for information about using the **ps** utility.
3. Check that the application is using the correct LU.
4. Check that the host link is active.
5. Ensure that the host activates the relevant LU.

Resolving MS Application Problems

To determine why a Management Services (MS) application is not working, issue the **snaadmin query_nmvt_application** command. Refer to *Communications Server for Linux Administration Command Reference* for more information about this command.

This command returns the following information:

- Name of the registered application
- MS vector key accepted by the application

Use the REGISTER_NMVT_APPLICATION verb to set the correct MS vector key. For more information, refer to the *Communications Server for Linux MS Programmer's Guide*.

Resolving NOF Application Problems

To determine why a Node Operator Facility (NOF) application is not working, obtain an API trace on the NOF interface. For more information about how to do this, see "API Tracing" on page 40.

Check that the application program has issued the SET_PROCESSING_MODE verb before issuing verbs that alter the configuration. Refer to the *Communications Server for Linux NOF Programmer's Guide* for more information about this verb.

Resolving Problems with TN Server

This section describes what to do if you encounter problems with TN Server for 3270.

Cannot Connect to the Host

If you cannot connect to the host using TN Server for 3270, then you should check the following:

1. Ensure that the node is started, and that the link to the host is in "active" or "on demand" state.
2. Ensure that the configuration contains a suitable TN3270 access record for the TN3270 user. Make sure the access record specifies the TN3270 client's address correctly or is the default record. Also make sure that the LU specified in the record is a valid LU or LU pool defined on the link to the host.

   Refer to the *Communications Server for Linux Administration Guide* for more information about configuring TN3270 users and LUs.
3. Ensure that the LU configured in the TN3270 access record is in SSCP state.
   - If the LU is inactive, you may be able to activate it by stopping and restarting the link. Ensure that no other users are using the link before you do this.
   - If the LU is already active, then it is in use by another user and cannot currently be used by this TN3270 client.
   - If the configuration specifies an LU pool rather than an individual LU, check that one or more LUs in the pool are in SSCP state. If all the LUs are active, this means that they are in use by other users and cannot currently be used by this TN3270 client.
4. If you use the Secure Sockets Layer feature of TN Server (SSL), check the SSL configuration:
   - The TN3270 emulator and the TN3270 access record must both be configured to use SSL, or must both be configured not to use SSL if you do not want to use it for this TN3270 client.
   - If you cannot configure the TN3270 access record to use SSL (the SSL option is disabled in the Motif Administration program, or SMIT or the administration command fails with the return code `FUNCTION_NOT_SUPPORTED`), this indicates that the software to support SSL is not installed on the server. Refer to *Communications Server for Linux Quick Beginnings* for more information on installing this software.
   - If the SSL software is installed correctly, and both the TN3270 emulator and the TN3270 access record are configured to use it, the security requirements for SSL may not be valid. One security requirement is that TN Server has an up-to-date certificate from a certificate authority accepted by the TN3270 emulator. This ensures that another program cannot intercept the TN3270 connection request and pretend to be a valid TN Server (because this program would not have the correct certificate and so would not be able to identify itself correctly to the emulator). To correct this, start the key management program **snakeyman** and follow the instructions in the online help; see *Communications Server for Linux Quick Beginnings* for more information.

Resolving Problems with TN Redirector

This section describes what to do if you encounter problems with TN Redirector.

Cannot Connect to the Host

If you cannot connect to the host using TN Redirector, then you should check the following:

1. Ensure that the node is started.
2. Ensure that the configuration contains a suitable TN Redirector access record for the TN Redirector user. Make sure the access record specifies the TN client's address correctly or is the default record, and that the TCP/IP port specified is the one being used by the client.

   Refer to the *Communications Server for Linux Administration Guide* for more information about configuring TN Redirector.
3. Ensure that the access record specifies the host's IP address correctly, and that the TCP/IP port specified is the one being used by the host.
4. Use the **ping** utility to check connectivity to the client's TCP/IP address. Repeat for the host.
5. If you use the Secure Sockets Layer feature of TN Server (SSL), check the SSL configuration:
   - The TN client and the TN Redirector access record must both be configured to use SSL, or must both be configured not to use SSL if you do not want to use it for this client.
   - If you cannot configure the TN Redirector access record to use SSL (the SSL option is disabled in the Motif Administration program, or SMIT or the administration command fails with the return code `FUNCTION_NOT_SUPPORTED`), this indicates that the software to support SSL is not installed on the server. Refer to *Communications Server for Linux Quick Beginnings* for more information on installing this software.
   - If the SSL software is installed correctly, and both the emulator and the access record are configured to use it, the security requirements for SSL may not be valid. One security requirement is that TN Redirector has an up-to-date certificate from a certificate authority accepted by the TN client. This ensures that another program cannot intercept the connection request and pretend to be a valid TN Redirector (because this program would not have the correct certificate and so would not be able to identify itself correctly to the emulator). To correct this, start the key management program **snakeyman** and follow the instructions in the online help; see *Communications Server for Linux Quick Beginnings* for more information.

Resolving Network Node Session Routing Problems

To determine why the network node is not routing sessions, do the following:

1. Issue the **snaadmin query_isr_sessions** to obtain current information about active sessions. Refer to *Communications Server for Linux Administration Command Reference* for more information about this command.
2. Check that the network IDs are the same on all machines in the network. Communications Server for Linux does not include border node support that is required to access machines with different network IDs.
3. Ensure that you are trying to route only independent APPC sessions using ISR (intermediate session routing). Other sessions can use SNA gateway or DLUR.

Resolving SNA Gateway Session Problems

To determine why SNA gateway sessions do not connect, check the following:

1. Check that the upstream and downstream link stations are active.
   - If you are using Motif administration program, zoom on the link stations.
   - If you are using the command-line administration program, issue the **snaadmin status_connectivity** command.
2. Check that the LUs are attached in the configuration and that they are not being used by other applications.
   - If you are using Motif administration program, zoom on the link stations and LU pool.
   - If you are using the command-line administration program, issue the **snaadmin query_downstream_lu** command.
3. If you are using a pool, check whether there are LUs available in the pool using one of the following ways:
   - If you are using Motif administration program, zoom on the link station and the LU pool.
   - If you are using the command-line administration program, issue the **snaadmin query_lu_0_to_3** command.

Resolving Server Administration Problems

To determine why servers cannot administer one another, do the following:

1. Check that one server is a master server in the domain.
2. Issue the **snaadmin query_sna_net** command on each server to obtain information about the configuration of the topology.
3. Check that the servers are in the same domain.
4. Check the TCP/IP connections using the Linux **ping** command.

Online Support Information

If you have followed the steps described in the previous sections and have not been able to resolve your problem, you may be able to find more information on the IBM web site. The Communications Server for Linux support page provides information about:

- Code fixes
- Helpful tips and techniques
- Newsgroups for discussions on Communications Server for Linux
- Support options

To access this information, use http://www.ibm.com/software/network/commserver/support.

Reporting Problems to Support Personnel

There are some system problems that you will not be able to resolve. In these cases, the *Message action* field may recommend that you contact your support personnel.

Types of Support Personnel

The following types of support personnel may be able to help you:

Support personnel for the remote system or for the network
Support personnel responsible for the SNA network and the remote systems with which Communications Server for Linux is communicating. For example, these people include the providers of the X.25 network (for X.25 problems), TCP/IP network personnel (for TN Server problems), host personnel (for or LUA problems), and the System Administrator of the remote system (for APPC or CPI-C problems).

Support services
IBM support personnel.

Information to Provide to Support Personnel

The more information you initially provide about your problem to your support team the more likely you are to receive a fast resolution. See the following sections for the types of information to collect.

Depending upon the nature and extent of the problem you report, support personnel may request that you run **snagetpd**, the command-line diagnostic collection utility. This utility automatically creates a file in compressed tar format that provides comprehensive data that they can use for diagnosing the problem. It includes all the information described here.

For more information about **snagetpd**, see Appendix C, "Using snagetpd," on page 69.

Readme File

Provide the following information in a readme file when submitting your problem to support personnel:

- A clear description of your problem. What does not work, or what does not work correctly? What did you expect to happen?
- The steps you took before the problem occurred.
- The time and date that the problem occurred.
- How often, if ever, you can reproduce the problem.
- Whether the function worked correctly in the past. If it did, what changes have occurred since it last worked?
- The message numbers and parameters of any messages written to the SNA log files that are related to the problem.

In addition to the information provided in the readme file, see "Other Information to Provide to Support Personnel."

Other Information to Provide to Support Personnel

In addition to the readme file, gather the following information so that you can make it available to support personnel.

Program Error Messages
If you have a problem that you cannot solve after reviewing the program error messages, do the following:

1. Note the message displayed on the screen.
2. Save the log files (see "Changing the Names and Locations of Log Files" on page 28).

Error Log and Trace File Information
Provide the files that you were using as the error and audit log files when

the error occurred (normally **/var/opt/ibm/sna/sna.err** and **/var/opt/ibm/sna/sna.aud**). If you were running with tracing enabled, also provide the trace files.

If you were running Communications Server for Linux with audit or exception logging disabled, attempt to reproduce the problem with all categories of logging enabled. If you can do this, provide the new log files (including all message categories).

System Configuration Information

If your support team asks for your configuration information, send them the following files:

- **sna_node.cfg** file, which stores node configuration information.
- **sna_domn.cfg** file, which stores domain configuration information.
- **sna_tps** file, which stores TP configuration information.

Software Version Information

If a problem cannot be resolved locally, your support team needs to know exactly what level of code is running on your machine. Use the appropriate Linux utility to display the overall version of the software package. All Communications Server for Linux code contains "tags" that identify the precise code level. Use the **snawhat** utility to extract this information. If a third party application has been linked with a static library, then the version of the library used can be determined by using **snawhat**.

To obtain version information about files, type the following on the command line:

snawhat *<filename>*

where *<filename>* represents the file or files for which you need version information.

For example, to obtain version information about:

- the static libraries used by a local directory file named **my_appl**, type the following on the command line:

  snawhat my_appl
- sna* executables installed on the local directory, type the following on the command line:

  snawhat sna*

System Resources

In addition to checking which programs are running, you can check the Linux processing environment. Your support team may ask you to run the **ps** standard Linux utility to obtain information about the status of Linux processes and resources.

Process Status (ps) Utility

The **ps** utility is a standard tool installed on Linux computers that provides information about the status of Linux processes on your machine. Run **ps** in the following situations:

- A program will not start.
- A program "hangs," crashes, or runs slowly.

- A program error message indicates that a running program is interfering with another process.

To obtain basic information about the Communications Server for Linux processes running, type the following on the command line:

ps -ef | fgrep sna

For more information about **ps**, refer to your Linux documentation.

Summary of Collecting Information for Support Personnel

This section summarizes the steps that you should take to collect information for support personnel if you have found a problem that you can reproduce.

Collecting Information for Support Personnel

If it is possible, you are recommended to delete all the existing diagnostic files before you start to collect diagnostic information. Because this requires stopping Communications Server for Linux, it may not be practical in some cases. In these cases, omit the first three steps below.

1. Stop Communications Server for Linux by issuing the command **sna stop**.
2. Delete the contents of /var/opt/ibm/sna, or move the contents to another directory if you need to keep them.
3. Restart Communications Server for Linux by issuing the command **sna start**.
4. Set the trace file size to a large value to ensure that all relevant trace information will be captured:

   snaadmin set_trace_file, trace_file_size = 2000000
5. Turn on audit and exception logging:

   snaadmin set_global_log_type, audit = YES, exception = YES, succinct_audits = NO, succinct_errors = NO
6. Turn on all tracing:

   snaadmin add_dlc_trace

   snaadmin set_trace_type, trace_flags = ALL
7. Follow the sequence of actions that reproduces the problem.
8. Turn off tracing:

   snaadmin remove_dlc_trace

   snaadmin set_trace_type, trace_flags = NONE
9. Run **snagetpd** to collect the log and trace information:

   snagetpd
10. Provide the snagetpd output to support personnel using whatever mechanism they recommend.

Chapter 3. Using Logging and Tracing

Logging and tracing are valuable diagnostic tools that provide you and your support team with useful information for solving Communications Server for Linux problems. This chapter describes how to perform logging and tracing using either the Motif administration program or the command-line administration program (refer to the *Communications Server for Linux Administration Command Reference* for complete descriptions of the commands). It also describes how to filter the information written to log files so that you do not record multiple instances of the same log message.

Samples of logging messages and tracing output are also provided.

Controlling Logging Using the Motif Administration Program

The easiest way to control logging of events for Communications Server for Linux is to use the Motif administration program (**xsnaadmin**). This program provides a graphical user interface from which you can perform diagnostic tasks, such as selecting the type of logging for your system, as well as the message types to log.

To start the Motif administration program, complete the following steps:

1. Type **xsnaadmin** at the command line and press **Enter**. The main window is displayed.
2. Select the **Diagnostics** menu.

   Note: You can also select the **Diagnostics** menu from the node window.
3. Select **Logging**.

The Logging dialog is displayed, which enables you to control the following logging activities:

Central logging or Local logging
: If you want a central error log for all servers and clients, choose **Central logging**. If you want logs to be made locally at each machine, choose **Local logging**. You can choose central or local logging only when the Motif administration program is in contact with a master server because central logging is performed by the master.

Log exceptions
: Select this option to instruct the system to log all exception events. If you make this selection, you will be prompted to select **verbose** or **succinct** logging. When you have done this, click **OK**.

  This selection affects all machines in the domain (unless they have local overrides configured using the command-line administration program).

Log audit messages
: Select this option to instruct the system to log all audit events. If you make this selection, you will be prompted to select **verbose** or **succinct** logging. When you have done this, click **OK**.

  This selection affects all machines in the domain (unless they have local overrides configured using the command-line administration program).

The Communications Server for Linux Motif administration program only sets the global log settings. Local log settings override the global settings and can be configured on a particular machine using the command-line administration program.

Controlling Logging Using the Command-Line Administration Program

You can use the command-line administration program to perform the following tasks.

- Change the names and locations of the log files and backup log files.
- Enable central or local logging.
- Enable or disable logging of audit and exception events on individual servers.
- Change the maximum log file size.
- Enable verbose or succinct logging.

Changing the Names and Locations of Log Files

Communications Server for Linux usually puts logs in two different files:

- Problems (always logged) and exceptions (if logged) are normally logged to the **/var/opt/ibm/sna/sna.err** file.
- Audits (if logged) are normally written to the **/var/opt/ibm/sna/sna.aud** file.

You can change the names and locations of these files using the command-line administration program in the following way:

1. Use the **snaadmin set_log_file** command to change the names of log files.
2. Use the **snaadmin query_log_file** command to check the current file name.

Refer to *Communications Server for Linux Administration Command Reference* for more information about issuing these commands.

You may find it easier to have all messages (error and audit) logged to the same file, to make it easier to see how they relate to each other. To do this, issue two **snaadmin set_log_file** commands, one with the *log_file_type* parameter set to `ERROR` and one with the parameter set to `AUDIT`. In both commands, specify the same file name.

For example, to record error log and audit log messages in a file named **sna.log**, specify the following commands:

```
snaadmin set_log_file, log_file_type = ERROR, file_name = sna.log
snaadmin set_log_file, log_file_type = AUDIT, file_name = sna.log
```

Audit and error log files are ASCII text files. Use a standard Linux text editor such as **vi** to view them.

Enabling Central Logging and Local Logging

In a client/server system, Communications Server for Linux sends all log messages to files on the master server by default (central logging). However, you can log messages for each server to files on that server (local logging).

If a server cannot locate the domain configuration file when it starts up (for example, because no master or backup server is active), it cannot determine whether to log centrally or locally and which server is the central logger. In this case, the server logs messages locally. When it later establishes contact with the

master server and determines that central logging is in use, it sends any further messages to the central logger and stops local logging.

You specify whether to log centrally or locally by using the **set_central_logging** command. To check the name of the server that is currently defined as the central logger (to which all log messages are sent) or to check whether central logging is currently enabled, use the **query_central_logger** and **query_central_logging** administration commands.

For more information about these administration commands, refer to the *Communications Server for Linux Administration Command Reference*.

Determining Which Messages Are Logged

Problem messages are always logged and cannot be disabled, but you can specify whether to log exception and audit messages. The initial default is to log exception messages but not audit messages. You can specify global settings for logging exception and audit messages on all servers by using the **snaadmin set_global_log_type** command. If necessary, you can override these settings for a particular server by using the **snaadmin set_log_type** command.

To determine what logging options are in effect issue the following:

- Use **snaadmin query_global_log_type** to check which categories of messages are recorded on servers that use the global settings.
- Use **snaadmin query_log_type** to check which categories of messages are being recorded on a particular server.

For more information about these administration commands, refer to the *Communications Server for Linux Administration Command Reference*.

Controlling Log File Size

Communications Server for Linux enables you to prevent log files from becoming too large and consuming disk resources. The **snaadmin set_log_file** administration command lets you specify the maximum size of a log file and the name of a backup file for each type of log information (audit or error). The default maximum file size is 1,000,000 bytes.

When a log file reaches the specified size, Communications Server for Linux renames the file to the name of the backup file (overwriting any existing backup log file) and then clears the log file. This means that the maximum amount of log information stored at any one time is twice the specified maximum file size (or four times the maximum file size if you are logging audit and error information to separate files).

You may need to increase the size of the log files to allow for the volume of log information (if your system is large enough). In particular, consider increasing the log file size to allow for the following:

- Large numbers of clients or users (because a single communications link failure can result in large numbers of logs on the server relating to session failures).
- Activation of audit logging as well as exception logging.
- Use of central logging instead of local logging.
- Use of verbose logging instead of succinct logging. For more information, see "Verbose Logging Message Format" on page 30.

The **snaadmin set_log_file** administration command also enables you to clear the current contents of the log file at any time (with or without copying the information to the backup file).

For more information, refer to the *Communications Server for Linux Administration Command Reference*.

Choosing the Format of Logs

You control the amount of detail recorded in logs by choosing one of the following logging formats:

Verbose logging
Each message contains the message number, originating component, type of message, text of the message, the cause of the condition leading to the message, and any recommended action.

Succinct logging
Each message contains only an abbreviated version of the header information (message number, originating component, and message type) and the text of the message. You can use the **snahelp** command-line utility to obtain the cause and action information for a particular message number (see "Using snahelp for Succinct Logging Messages" on page 33).

The default is succinct logging. You can specify verbose logging for audit messages, for error messages (problem and exception), or for both message types by using the **set_global_log_type** commandor **snaadmin set_log_type** command. If you are using central logging, the choice of succinct or verbose logging is determined by the settings on the server acting as the central logger, so all messages of the same type (audit or error) are written to the file in the same format.

When using succinct logging, a few messages from the API components may still be written to the log file in verbose format. This generally occurs when Communications Server for Linux is terminating or experiencing certain error conditions; in these cases, the component cannot obtain information about the node's configuration and cannot determine the log format to use.

Verbose Logging Message Format

The following example shows a typical log message in verbose logging format. The text following the example explains the items in the message.

```
Verbose Logging Message Format
---------------------- 13:55:16 EDT  15 May 1997 ----------------------
CFG_DAEM Message 4097 - 132, Subcode: 1 - 1
Log category: PROBLEM   Cause Type: External
System: sna18
Process ID: 17908 (snacfgdae)

The initially active port could not be started when starting the node.
Port name = SDLCP0Cause:  The config daemon could not start the port while loading the node's
configuration. The node will be started, but the port will not be started.
Action:  Check for other logs which indicate why the port failed to start.
Check that the DLC has been started.
```

Verbose logging messages contain the following information:

Timestamp
The time and date the message was generated. If you are using central logging, the timestamp for each message is taken from the system clock on

the computer where the message was generated. If the system clocks on different computers are not synchronized (because of clock inaccuracies or time zone differences), the messages in the central log file may appear out of sequence because they are added to the file in order of arrival at the central logger and not in order of their timestamps.

If your system includes Windows clients and you are using central logging, ensure that the `TZ` entry in the Windows Registry is set to indicate the correct time zone. If this is not set correctly, timestamps for logs from the Windows client may be incorrect. Refer to your Windows documentation for more information about the setting of `TZ`.

Component
The Communications Server for Linux component (such as local node, link driver, or APPC library) that logged the message.

Message number
An identifier for the message. This identifier consists of two numbers.

Subcode
A unique identifier that indicates the point within Communications Server for Linux at which the message was logged. This subcode is used only by Communications Server for Linux support personnel.

Log category
The event category of the log message. Possible values are `Problem`, `Exception`, or `Audit`.

Cause type
The cause of the message. Possible values are:

`Internal`
Internal error in a Communications Server for Linux component. Report errors of this type to Communications Server for Linux support personnel.

`System limit`
An internal limiting value (for example, entries in a fixed-size table) in the Communications Server for Linux software. There are very few instances when this type of log occurs.

`External`
A cause external to Communications Server for Linux (such as a problem with communication link hardware) or in non-Communications Server for Linux software (such as communication link drivers).

`Resource`
Resource shortage (for example, insufficient memory on the Linux computer).

`User` User error (for example, invalid parameters supplied on the command line to a Communications Server for Linux program).

`SNA` SNA protocol violation by a remote system, or interoperability problem with another SNA implementation.

`Config` Error in the Communications Server for Linux configuration, or mismatch between the Communications Server for Linux configuration and the remote system.

`Audit` A normal event, reported for information only.

System name
The name of the computer where the condition that caused the message was detected.

Process ID and name
The Linux process ID (from the computer whose system name is shown) and the executable name of the process that logged the message. The process ID is shown only for user-space components. For a message logged by a Windows client, this parameter identifies the Windows task handle of the process.

Message text
Text describing the condition being logged. This field may include a number of variable parameters relating to this particular occurrence of the message. For example, a message reporting startup of an APPC session may include the names of the local and partner LUs and the mode they are using for this session.

This field can contain the return code from an operating system call. For a message logged by the Linux computer, it may be shown either as a symbolic name or as a numeric value. Check the numeric values in the **/usr/include/sys/errno.h** file on the computer where the error occurred to find the corresponding symbolic name. The symbolic names are listed in your operating system documentation.

For a message logged by a Windows client, refer to your Windows documentation for explanations of these return codes.

Message cause
Additional information about the cause of the condition being logged. This field may not be included if the message text contains all the required information. This field is generally not used when the cause type is `Internal`.

Message action
Recommended action as a result of the message. For audit messages, which provide accounting and progress information instead of reporting error conditions, this field is not included because no action is required.

Succinct Logging Message Format

The following shows the succinct logging format:

```
13:55:16 EDT  15 May 1997  4097-132 (1-1)  P  sna18     PID 17908 (snacfgdae)
The initially active port could not be started when starting the node.
Port name = SDLCP0
```

The first line contains all of the following fields except the message text, which appears on the second line:

Timestamp
The time and date the message was logged.

Message number
An identifier for the message. This identifier consists of two numbers separated by a hyphen (-).

Subcode
A unique identifier that indicates the point within Communications Server for Linux at which the message was logged, shown in parentheses after the message number. This subcode is used only by Communications Server for Linux support personnel.

Log category
The category of the log message, shown as a single character:
- `P` (problem)
- `E` (exception)
- `A` (audit)

System name
The name of the computer where the condition that caused the message was detected.

Process ID and name
The Linux process ID (preceded by the characters *PID*) and executable name of the process that logged the message. The process ID is shown only for the API components. For a message logged by a Windows client, this parameter identifies the Windows task handle of the process.

Message text
Text describing the condition being logged. This field may include a number of variable parameters relating to this particular occurrence of the message. For example, a message reporting startup of an APPC session may include the names of the local and partner LUs and the mode they are using for this session.

This field can contain the return code from an operating system call. For a message logged by the Linux computer, it may be shown either as a symbolic name or as a numeric value. Check the numeric values in the **/usr/include/sys/errno.h** file on the computer where the error occurred to find the corresponding symbolic name. The symbolic names are listed in your operating system documentation.

For a message logged by a Windows client, refer to your Windows documentation for explanations of these return codes.

Using snahelp for Succinct Logging Messages

The succinct logging format does not show cause and action information. However, you can use the **snahelp** utility to obtain details about the cause and action for a particular message number by typing the following command at the Linux command prompt:

snahelp *message_number*

The *message_number* is returned by Communications Server for Linux in the header information for the message and consists of two numbers separated by a hyphen (**-**).

The utility returns the name of the component that logged the message and the information from the *Message number*, *Cause type*, *Message cause*, and *Message action* fields as shown for verbose logging.

For example, to obtain further information about the succinct log message shown in "Succinct Logging Message Format" on page 32 (with component ID and message number 4097 - 132), type the following command:

snahelp 4097-132

The output from **snahelp** is:

```
snahelp Output
CFG_DAEM Message: 4097 - 132, Cause Type: External

Cause:  The config daemon could not start the port while loading the node's
configuration. The node will be started, but the port will not be started.
Action:  Check for other logs which indicate why the port failed to start.
Check that the DLC has been started.
```

Filtering Logging

If you find that a particular event is occurring frequently and so the log file is filling up with many instances of the same log message, you can set a filter to specify that one or more specific log messages should be logged only once. Any subsequent instances of the same log message will be ignored and will not be written to the log file. This filtering applies to all types of logs: audit, exception, and problem logs.

If you are using central logging, you set up the filter on the server that is acting as the central logger. This means that the message will be written to file only once, even if it occurs on two or more different servers. You are recommended to duplicate the filter on any other servers that can act as backup master servers, so that the filtering will continue if the central logger is stopped and another server takes over. In addition, you can set this filter on a Remote API Client; this means that the specified message will be sent to the central logger only once from this client.

If you are not using central logging, you can set the filter on each server or Remote API Client, so that the message will be logged only once on that system. There is no need to set up the same filtering options on all servers and clients; you can set the filter or leave it unset on each server or client according to your requirements..

To set up a filter for one or more log messages, create an ASCII file **logfilter.txt** in the following directory:

- On Linux servers or Linux Remote API Clients: **/etc/opt/ibm/sna**
- On AIX Remote API Clients: **/etc/sna**
- On Windows Remote API Clients: the directory specified in the *Logging* / *log_directory* parameter in the Windows Registry. If this parameter is not specified, the default is the Remote API Client's install directory (for example **c:\ibmcs\w32cli**).

Each line in this file contains the message number of a particular log message that you want to filter. Specify this as two numbers separated by a hyphen (dash) character, as described in "Verbose Logging Message Format" on page 30. For example:

```
1024-15
2048-12
512-16
```

You can include a maximum of 20 message numbers in this file. If you include more, only the message numbers specified in the first 20 lines of the file are filtered, and the remaining lines in the file are ignored.

This file is read when the Communications Server for Linux software is started on the server or client containing the file.

- If you create a new **logfilter.txt** file or modify the existing file, you will need to stop and restart the Communications Server for Linux software on the server or client containing the file for your changes to take effect.
- The count for each filtered log message is reset when the Communications Server for Linux software is stopped and restarted, or when the server acting as the central logger is stopped and another server takes over. This means that the message is logged only once for each run of the Communications Server for Linux software or each time a server takes over as the central logger.

Usage Logging

You may want to keep track of Communications Server for Linux resource usage, such as the numbers of links and sessions that are active on the local node at any time. In particular, you need to use this information to ensure that your usage of Communications Server for Linux resources is within the limits permitted by your license. For more information about licensing requirements, see *Communications Server for Linux Quick Beginnings*.

Communications Server for Linux provides two methods for accessing this information:

- Every 30 minutes, details of the current usage and the peak usage (the maximum usage level at any time since the Linux computer was restarted) are written to a file known as the usage log file. You can review the contents of this file to see how the usage changes over time.
- You can get a "snapshot" of the current and peak usage at any time using the administration command **query_node_limits** or the NOF verb QUERY_NODE_LIMITS.

Usage Log File

Communications Server for Linux usually records usage information in the file **/var/opt/ibm/sna/sna.usage**. When this file reaches 1,000,000 bytes, Communications Server for Linux renames the file to **/var/opt/ibm/sna/bak.usage** (overwriting any existing backup usage log file) and then clears the log file. This means that the maximum amount of log information stored at any one time is twice the specified maximum file size.

You can use the command-line administration program to change the name and location of the usage log file, or the maximum file size, in the same way as for audit and error log files. You can also clear the current contents of the log file at any time (with or without copying the information to the backup file). Use the following commands:

1. Use the **snaadmin set_log_file** command to change the log file name or maximum size, or to clear the file.
2. Use the **snaadmin query_log_file** command to check the current usage log file settings.

Refer to *Communications Server for Linux Administration Command Reference* for more information about issuing these commands.

The format of the usage log file is as follows:

- The file is divided into a number of columns, each recording usage of a particular resource type:
  - APPC and CPI-C applications

  - LUA applications
  - Active link stations
  - TN3270 sessions using TN Server
  - Telnet sessions using TN Redirector
  - Data sessions (PLU-SLU sessions)
- Each column shows two figures: the current usage of a particular resource type at the time it was recorded, and the peak usage (the maximum usage level of the resource type at any time since the Linux computer was restarted).
- Each line in the file represents a "snapshot" of the resource usage at a particular time, shown by a timestamp at the end of the line. Usage is recorded at 30–minute intervals.

Using Administration Tools to Check Resource Usage

To check resource usage at a particular time, you can use the administration command **query_node_limits** or the NOF verb QUERY_NODE_LIMITS. Refer to *Communications Server for Linux Administration Command Reference* or *Communications Server for Linux NOF Programmer's Guide* for more details.

The command or verb returns information about the same resource types as in the usage log file, giving the current and maximum usage of each resource type. It also returns information about the functions that your Communications Server for Linux license allows you to use.

Tracing

This section explains how to use Communications Server for Linux trace facilities to collect diagnostic data while the Communications Server for Linux system is running and how to produce trace output.

Note: Turn tracing on only when you are requested to do so by support personnel or when you need the trace output to diagnose a problem. At other times, turn off all tracing because it degrades system performance.

You can use either the Motif administration program or the command-line administration program to do the following:

- Enable or disable line tracing for various link types.
- Enable or disable client/server tracing.
- Enable or disable internal tracing.

API tracing can be enabled or disabled for the APPC, CPI-C, LUA, MS, CSV, and NOF APIs. Usually you set up API tracing using the `SNATRC` environment variable, but occasionally you may need to use a combination of environment variables, verbs, and program functions to do this.

Figure 1 on page 37, shows the interfaces where each of the main types of tracing occurs in the overall structure of Communications Server for Linux.

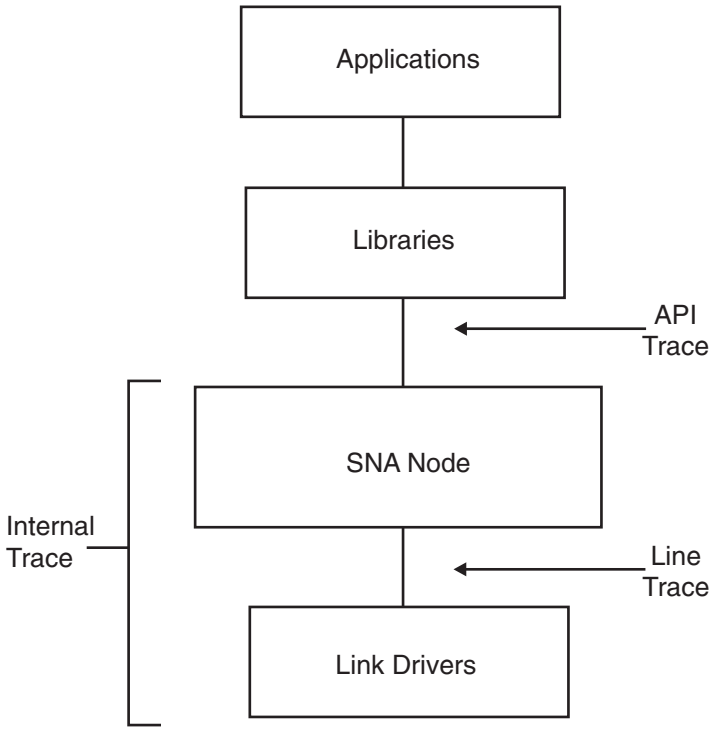

Figure 1. Tracing Interfaces

The trace facilities write data to text files and to binary files. Use any standard ASCII text editor to view the text files. Use the **snatrcfmt** command-line utility to convert the binary output to text files so that you can also view them with an ASCII text editor. For more information, see "Formatting Internal Binary Trace Output into Text Files" on page 57.

The following sections describe each type of tracing Communications Server for Linux provides (listed in order of usefulness to end-users, beginning with the most useful).

Line Tracing

The type of tracing you are most likely to need is line tracing, which traces SNA messages flowing on the communications link to the remote system. The following section explains how to perform line tracing using the Motif administration program and the command-line administration program.

Performing Line Tracing

To perform line tracing using the Motif administration program, complete the following steps:

1. Enter **xsnaadmin** from the command line to start the Motif administration program and display the main window.
2. Select the **Diagnostics** menu, then select **Node tracing**. The Tracing dialog is displayed.
3. Select one of the following DLCs to perform line tracing for:
   - **Token Ring**
   - **Ethernet**

- **SDLC**
- **X25**
- **MPC Channel links** (Multipath Channel)
- **Enterprise Extender links**

4. Optionally, select **Truncate messages** and enter the maximum message size, in bytes. This can be helpful if a large amount of data is being traced and you are interested in the protocol exchanges rather than in the data itself. Protocol data is usually at the start of the individual messages.
5. When you have finished, click **OK**.

If you are tracing an SDLC line and would like more detailed trace information, you can get this by using internal tracing on SDLC as well as line tracing. The additional detail is formatted as part of the output for line tracing, so that you will see all of the SDLC tracing in one file. For more information, see "Internal Tracing" on page 50.

Tracing is now enabled and written to the following binary files:

- **/var/opt/ibm/sna/sna1.trc**
- **/var/opt/ibm/sna/sna2.trc**

To perform line tracing using the command-line administration program, complete the following steps:

1. If you do not wish to use the default settings then use the **snaadmin set_trace_file** command to specify attributes of the tracing file or files:
   - Add the *dual_files* parameter to specify whether tracing is to one or two files:
     - To trace to two files, specify

       snaadmin set_trace_file, dual_files = YES
     - To trace to one file, specify

       snaadmin set_trace_file, dual_files = NO
   - Add the *trace_file_size* parameter to specify the maximum size of the trace file, in bytes. (If *dual_files* is set to `NO`, this parameter is ignored.) For example:

     snaadmin set_trace_file, trace_file_size = 1000000
   - Add the *file_name* and *file_name_2* parameters to rename the trace files. (If *dual_files* is set to `NO`, the *file_name_2* parameter is ignored.) For example:

     snaadmin set_trace_file, file_name = new1.trc, file_name_2 = new2.trc
2. Perform one of the following procedures:
   - Activate line tracing on all DLCs, ports, and link stations (LSs) by specifying the following:

     snaadmin add_dlc_trace
   - Control the amount of line tracing by specifying the resource type (for example, port and link station):
     - To restrict the tracing to a particular port, specify the command with the following parameters:

snaadmin add_dlc_trace, resource_type = PORT, resource_name = *port_name*

– To restrict the tracing to a particular LS, specify the command with the following parameters:

snaadmin add_dlc_trace, resource_type = LS, resource_name = *LS_name*

Tracing is now enabled and written to the following binary files (unless the trace files have been renamed):

- **/var/opt/ibm/sna/sna1.trc**
- **/var/opt/ibm/sna/sna2.trc**

For more information about the **snaadmin add_dlc_trace** command, refer to the *Communications Server for Linux Administration Command Reference*.

If you are tracing an SDLC line and would like more detailed trace information, you can get this by using internal tracing on SDLC as well as line tracing. The additional detail is formatted as part of the output for line tracing, so that you will see all of the SDLC tracing in one file. For more information, see "Internal Tracing" on page 50.

Formatting the Binary Trace File

The **snaadmin add_dlc_trace** command generates a binary trace file or files containing only line trace messages. The trace data is stored in the files **/var/opt/ibm/sna/sna1.trc** and **/var/opt/ibm/sna/sna2.trc** (unless you used the **snaadmin set_trace_file** administration command to specify a different trace file or files).

To change the binary files into ASCII text output, change to the directory **/var/opt/ibm/sna** and use the **snatrcfmt** utility with its default options on each of the trace files.

For example, specify the following:

snatrcfmt -f sna1.trc -o sna1

snatrcfmt -f sna2.trc -o sna2

If you specified different trace files, replace **sna1.trc** and **sna2.trc** with the appropriate file names. For more information about **snatrcfmt**, see "Formatting Internal Binary Trace Output into Text Files" on page 57.

Note: If the data flowing to and from the remote system is compressed (as determined by the configuration of the APPC mode, LS or internal PU that the session uses), the trace formatter does not uncompress this data or translate it to ASCII.

Use an ASCII text editor to view the message data dump files, **sna1.dmp** and **sna2.dmp**. These files contain the SNA message data; each message is listed as hexadecimal data, and is interpreted as EBCDIC and as ASCII. In addition, the *TH* and *RH* fields in the message header are interpreted as text strings preceding the message data.

The corresponding message flow drawing files, **sna1.drw** and **sna2.drw**, are not created. There is no message flow drawing because the data being traced is flowing from Communications Server for Linux to a remote component (which is not shown in the diagram) rather than between two Communications Server for Linux components.

The following is a sample line trace file.

```
Sample Line Trace File
------------------------------------------------ 17:21:04.040 PDT  22 Oct 1997
SND>> ACTPU   +RSP                      LFSID:00000 TOKR0.TOKRP1.TOKRL1
TH:   2D000000BC9B BBIU EBIU EFI      OAF:00 DAF:00 SNF:BC9B
RH:   EB8000       SC  FI
RU:   11124040 40404040 40400000 07010000   ..        ...... ..@@@@@@@@......
      00000000                              ....              ....
------------------------------------------------ 17:21:04.230 PDT  22 Oct 1997
SND>> ACTLU   +RSP                      LFSID:02000 TOKR0.TOKRP1.TOKRL1
TH:   2D000002BC9C BBIU EBIU EFI      OAF:02 DAF:00 SNF:BC9C
RH:   EB8000       SC  FI
RU:   0D020100 85800000 0C060100 01000000   ....e........... ................
------------------------------------------------ 17:21:04.240 PDT  22 Oct 1997
SND>> NOTIFY  RQD1                      LFSID:02000 TOKR0.TOKRP1.TOKRL1
TH:   2C0000020000 BBIU EBIU          OAF:02 DAF:00 SNF:0000
RH:   0B8000       FMD FI                        BC EC
RU:   8106200C 06030001 000000              a..........      .. ........
------------------------------------------------ 17:21:04.460 PDT  22 Oct 1997
<<RCV FMD     +RSP                      LFSID:02000 TOKR0.TOKRP1.TOKRL1
TH:   2C0000020001 BBIU EBIU          OAF:02 DAF:00 SNF:0001
RH:   838000       FMD
------------------------------------------------ 17:21:04.550 PDT  22 Oct 1997
```

API Tracing

API tracing can help you locate communication problems that involve programs using APIs. You cannot use the Motif administration program or the command-line administration program to enable or disable API tracing. In most cases, you use the `SNATRC` environment variable to set up tracing. If you want to be able to control tracing while the application is running, you may need to use a combination of environment variables, verbs, and program functions.

Communications Server for Linux supports tracing for the following APIs:

- APPC
- CPI-C
- LUA
- MS
- CSV
- NOF

For the APPC, CSV, LUA, MS, and NOF APIs, which use Verb Control Blocks (VCBs), the trace file includes a dump of the VCB contents (in hexadecimal, interpreted as ASCII, and interpreted as EBCDIC). If you want a more detailed interpretation of the VCB contents, you can use the API trace formatter utility **snaapitrcfmt** to process the trace file; see "API Trace Formatter" on page 45 for more information. The **snaapitrcfmt** utility has no effect on CPI-C tracing, because CPI-C uses function calls instead of VCBs; the individual parameters are already shown in the trace file and do not need further interpretation.

Performing API Tracing

To set up and control API tracing, complete the following steps before starting the application program using the API:

1. Use the `SNATRC` environment variable to specify one or two trace data files and whether to activate tracing when the application starts. The syntax for the `SNATRC` environment variable is:

   file1 [**:**[*file2*][**:**]]

Specify the following parameters:

file1 Name of the trace file.

file2 Optionally included name of a second trace file. Use a colon to separate the two file names.

Final colon (:)
The optional final colon indicates that tracing is active as soon as the application is started. If tracing is not active when the application is started, activate it from within the application by using the CSV DEFINE_TRACE verb. If you specify only one file name, use two colons to make tracing active as soon as the application is started.

For example:

- If you type **export SNATRC=file1.trc:file2.trc** at the command line, tracing is to two files, and tracing is inactive when the API application is started. However, you can start it later, as described in "Controlling API Tracing from within an Application."
- If you type **export SNATRC=file.trc::** at the command line, tracing is to one file, and tracing starts when the API application starts.

2. If you specify two file names, use the SNATRACESIZE environment variable to set the maximum size of each trace file. (If you specify only one file name, the size of the trace file has no limit.) For more information about how to use SNATRACESIZE, see "Controlling Trace File Contents" on page 42.

   Specify a path for the text file to which trace data will be written. Do not specify the name of a device (such as **/dev/tty)** or a print spooler as a trace file name. If you do not specify a full path for the trace file or files, Communications Server for Linux uses the directory from which you start the application.

Controlling API Tracing from within an Application

Communications Server for Linux provides the ability to control API tracing from within an application program. You can trace the specific section of a program where a problem is being encountered without having to trace the whole application. Tracing can be activated at the start of the section and deactivated at the end of it.

To use this facility, complete the following steps:

1. Set up the SNATRC environment variable before starting the application program (see "Performing API Tracing" on page 40). When setting this environment variable, you can specify tracing to be active or inactive when the application starts. You can then activate it or deactivate it from within the application as required.
2. Control the tracing (that is, enable or disable tracing) within the program by using the CSV DEFINE_TRACE verb (for more information, refer to the *Communications Server for Linux CSV Programmer's Guide*).

Disabling the Application's Control of Tracing

The SNACTL environment variable overrides the ability of application programs to control API tracing. You can use SNACTL to force the tracing of an entire application program that normally turns off tracing for some functions or to prevent tracing for an application program that normally uses it. When SNACTL is set before the application program is started, any trace control commands issued by application programs are ignored. If tracing is on, it remains on; if tracing is off, it remains off. However, the return code from any issued trace control command indicates successful completion.

To use SNACTL, set it to any string (such as **export SNACTL=1**). To cancel SNACTL, set it to a null string.

Controlling Tracing on Automatically Started Invokable TPs

You can trace an automatically started invokable transaction program (TP) by configuring the appropriate environment variables when you run the **snatpinstall** program. Refer to the *Communications Server for Linux Administration Guide* for more information.

Controlling Trace File Contents

The following environment variables control the amount of data stored in trace files:

SNATRUNC

Specifies the maximum length in bytes of each entry in a trace file. Set this variable to a decimal number. If a message has more characters than this value, the excess characters are truncated. For example, setting SNATRUNC to 70 limits tracing to 70 bytes of data per entry. By default, API trace messages are not truncated.

SNATRACESIZE

Specifies the maximum size in bytes of each trace file when using two files. If you are tracing to one file, the size of the trace file has no limit.

Set this environment variable to a decimal number. When the size of **file1** reaches the maximum file size, Communications Server for Linux clears **file2** and continues tracing to **file2**. When **file2** reaches the same limit, Communications Server for Linux clears **file1** and writes the trace information to **file1** again. This ensures that the maximum amount of disk space taken up by a pair of API trace files is approximately twice the value of SNATRACESIZE. Using two files extends the trace period and limits the disk space usage to twice the value specified in SNATRACESIZE.

If you do not set SNATRACESIZE, the default is 1,000,000 bytes. To cancel the setting of SNATRACESIZE and return to the default, set SNATRACESIZE to a null string.

SNATRCRESET

Controls whether a trace file is reset when an application first writes to it. Normally, the file is reset and its contents discarded when an application writes its first trace message to the file. If you are tracing two or more applications to the same file, or if you want to trace two or more runs of the same application to the same file, you can prevent the file from being reset by setting the SNATRCRESET environment variable to NO.

If you are tracing to two files, the files continue to be reset as normal when the maximum file size is reached, but they will not be reset when an application starts tracing for the first time. If you are tracing to one file, setting SNATRCRESET to NO means that the file will never be reset automatically. To avoid taking up too much disk space, delete it manually from time to time.

To cancel the setting of SNATRCRESET and return to the default setting so that the file is reset when an application first traces to it, set SNATRCRESET to a null string.

Trace File Format for API Tracing

The trace data for a single message can occupy more than one line in the trace file. Each individual message is preceded by a horizontal line indicating the time the trace entry was made. The following describes a Communications Server for Linux API trace file:

- The process ID of the component being traced appears at the start of each line. After the process id is the thread id (separated by a period). For a single-threaded application, this will always be 00. The process ID is followed by an indicator of the component type being traced (for example, APPC, for an APPC TP).
- Message data is shown in the following formats in separate columns to ensure that a character string in the message data appears as readable text in either the EBCDIC or the ASCII column, according to its character set:
  - Hexadecimal
  - Interpreted as EBCDIC
  - Interpreted as ASCII

The format of trace data varies slightly among the APIs:

APPC, NOF, MS
: The verb control block supplied by the application to the corresponding API library is traced when the verb is issued and when it returns. For verbs issued through the asynchronous entry point and for the APPC [MC_]RECEIVE_AND_POST verb issued using the synchronous entry point, both the initial return (indicating that the verb was issued successfully and is in progress) and the return to the callback routine (when the verb completes) are traced.

  The top of each section of tracing shows the name of the verb issued and the result on its return (taken from the primary return code). Any data being sent or returned is also traced following the verb control block (VCB), with the address at which the data is stored (taken from the verb control block).

  If an MS or NOF application has registered to receive indications, each indication is also traced in a format similar to an asynchronous verb return.

  For information about the VCB structures and content for each of these APIs, refer to the *Communications Server for Linux APPC Programmer's Guide*, *Communications Server for Linux NOF Programmer's Guide*, and *Communications Server for Linux MS Programmer's Guide*.

CSV
: The verb control block supplied to the CSV library is traced when the verb is issued and when it returns. The top of each section of tracing shows the name of the verb issued and the result on its return (taken from the primary return code). Any data string included in a verb is also traced following the verb control block with the address at which the data is stored (taken from the verb control block).

  For more information, refer to the *Communications Server for Linux CSV Programmer's Guide*.

LUA
: The verb control block supplied to the LUA library is traced when the verb is issued and again when it returns.

  If the verb returns asynchronously (the *lua_flag2.async* bit is set to 1 and the primary return code is set to `LUA_IN_PROGRESS`), the verb control block is traced a third time when it completes. In this case, you should ignore the

parameters in the initial return VCB (except for *lua_flag2.async* set to 1 and the primary return code set to LUA_IN_PROGRESS, which indicate that this is an asynchronous verb return) and take account only of those in the final return VCB when the verb completes. In this VCB, *lua_flag2.async* remains set to 1 and the primary return code is set to a value other thanLUA_IN_PROGRESS.

The top of each section of tracing shows the name of the LUA verb issued and the result on its return (taken from the primary and secondary return codes). Any data being sent or returned is also traced following the verb control block, with the address at which the data is stored (taken from the verb control block).

Because Communications Server for Linux implements the SLI using RUI verbs, the LUA library converts SLI verbs into the corresponding RUI verbs (where each SLI verb may result in one or more RUI verbs). Therefore, SLI tracing includes both SLI parameters and RUI parameters. First, the SLI request is traced, then the RUI request and return for each verb (including the later asynchronous return if applicable), and finally the SLI return.

For more information, refer to the *Communications Server for Linux LUA Programmer's Guide*.

CPI-C Because Communications Server for Linux implements CPI-C using APPC, the CPI-C library converts most CPI-C calls into the corresponding APPC verbs. Therefore, CPI-C tracing includes both the CPI-C parameters and the APPC parameters. First, the CPI-C request is traced, then the APPC request, then the APPC return, and finally the CPI-C return. For other CPI-C functions that deal only with local information (such as checking or setting the receive type or the synchronization level), no APPC verbs are executed, so the tracing shows only the CPI-C parameters.

The top of each section of tracing shows the name of the CPI-C call or APPC verb issued and its return code. Any data being sent or returned is also traced following the CPI-C parameters or APPC verb control block, with the address at which the data is stored.

For more information, refer to the *Communications Server for Linux CPI-C Programmer's Guide*.

The following examples show fragments of API trace files from CPI-C and APPC applications:

```
Sample API Trace File Fragment: CPI-C
|================================================================
|=============== Initialized 14:40:35 BST  15 Sep 1997 ==========
|================================================================
 4849.00 CPIC   ------------------------------ 14:40:35.07 BST  15 Sep 1997
 4849.00 CPIC   CMINIT request
 4849.00 CPIC    Sym dest name =
 4849.00 CPIC   ------------------------------ 14:40:35.08 BST  15 Sep 1997
 4849.00 CPIC   CMINIT response, result = CM_OK
 4849.00 CPIC    Conversation ID = 01000001
 4849.00 CPIC   Conversation characteristics
 4849.00 CPIC    Conversation type = CM_MAPPED_CONVERSATION
 4849.00 CPIC    Deallocate type = CM_DEALLOCATE_SYNC_LEVEL
 4849.00 CPIC    Error direction = CM_RECEIVE_ERROR
 4849.00 CPIC    Sync level = CM_NONE
 4849.00 CPIC    Fill type = CM_FILL_LL
 4849.00 CPIC    Prepare to receive type = CM_PREP_TO_RECEIVE_SYNC_LEVEL
 4849.00 CPIC    Receive type = CM_RECEIVE_AND_WAIT
 4849.00 CPIC    Send type = CM_BUFFER_DATA
 4849.00 CPIC    Conversation security type = XC_SECURITY_SAME
 4849.00 CPIC    Log data pointer = 0
 4849.00 CPIC    Log data length = 0
 4849.00 CPIC    Sym dest name =
 4849.00 CPIC    Partner LU name =
 4849.00 CPIC     20202020 20202020 20202020 20202020     .................
 4849.00 CPIC     20                                      .@@@@@@@@@@@@@@@@
 4849.00 CPIC    Mode name =
 4849.00 CPIC     40404040 40404040                       @@@@@@@@@@@@@@@@
 4849.00 CPIC    Partner TP name =
 4849.00 CPIC     40404040 40404040 40404040 40404040     @@@@@@@@@@@@@@@@
 4849.00 CPIC     40404040 40404040 40404040 40404040     @@@@@@@@@@@@@@@@
 4849.00 CPIC     40404040 40404040 40404040 40404040     @@@@@@@@@@@@@@@@
 4849.00 CPIC     40404040 40404040 40404040 40404040     @@@@@@@@@@@@@@@@

Sample API Trace Fragment: APPC
 2511.00 APPC   ------------------------------ 14:49:08.04 BST  20 Oct 1998
 2511.00 APPC   TP_STARTED request
 2511.00 APPC   ---- Verb Parameter Block at address 40001578 ----
 2511.00 APPC     00140000 00000000 00000000 54504C55     .............&<.  .............TPLU
 2511.00 APPC     31202020 00000000 00000000 E3D7D5C1     ............TPNA  1   ............
 2511.00 APPC     D4C5F140 40404040 40404040 40404040     ME1               ...@@@@@@@@@@@@@
 2511.00 APPC     40404040 40404040 40404040 40404040                       @@@@@@@@@@@@@@@@
 2511.00 APPC     40404040 40404040 40404040 40404040                       @@@@@@@@@@@@@@@@
 2511.00 APPC     40404040 40404040 40404040 00000000                  ....  @@@@@@@@@@@@....
```

For the APPC, CSV, LUA, MS, and NOF APIs, you can use the API trace formatter utility **snaapitrcfmt** (**snaapitrcfmt64** for a 64–bit Linux application) to provide a more detailed interpretation of the VCB contents. See “API Trace Formatter” for more information.

API Trace Formatter

The **snaapitrcfmt** command-line utility provides a more detailed expansion of VCB contents in APPC, CSV, LUA, MS, and NOF trace files, by interpreting the contents of each parameter within the VCB and presenting it as plain text. It takes a standard Communications Server for Linux API trace file as input, and writes the detailed trace expansion to a new text file.

The **snaapitrcfmt** utility does not expand CPI-C tracing, because the function parameters for CPI-C are already interpreted in the standard trace file format. However, any CPI-C tracing in the input file is written to the output file unchanged, to ensure that it is not lost if the input file contains more than one trace type. In particular, because Communications Server for Linux CPI-C is implemented over the APPC interface, a CPI-C trace file also contains tracing for the underlying APPC VCBs; if you use the **snaapitrcfmt** utility to process a CPI-C trace file, the output file contains the original CPI-C tracing unchanged with a detailed expansion of the APPC VCBs.

The syntax of the command to run the API trace format utility is:

snaapitrcfmt [**-f** *source_file_1*[**:***source_file_2*]] [**-o** *output_file*] [
-h]

Specify the following options and parameters:

-f *source_filenames*
Use this option to specify the name of the input API trace file or files. If the SNATRC environment variable is currently set to the name of the correct input file or files, you do not need to specify this option.

- If the tracing you want to format is in a single file, for example myapi.trc, use the following format:

  -f myapi.trc
- If the tracing you want to format is in a pair of trace files, for example myapi1.trc and myapi2.trc, use the following format:

  -f myapi1.trc:myapi2.trc

If you do not specify a full path for the trace file or files, **snaapitrcfmt** uses the directory from which you start the application. If you do not use the **-f** option, **snaapitrcfmt** uses the file or files specified by the SNATRC environment variable. If SNATRC is not set, the utility uses **snaapi.trc** (in the current directory) as the default.

-o *output_file*
The name of the output file to be created by **snaapitrcfmt**. If the file already exists, its contents are replaced by the output from **snaapitrcfmt**.

If you do not specify a full path for the output file, **snaapitrcfmt** uses the directory from which you start the application. If you do not use the **-o** option, **snaapitrcfmt** uses **snaapi.dmp** (in the current directory) as the output file.

-h Displays help information for the **snaapitrcfmt** utility.

The following example shows a fragment of the output for an APPC trace file:

```
Sample Formatted API Trace Fragment: APPC
| 2511.00 APPC   ------------------------------- 14:49:08.04 BST  20 Oct 1998
| 2511.00 APPC   TP_STARTED request
| 2511.00 APPC   ---- Verb Parameter Block at address 40001578 ----
| 2511.00 APPC     00140000 00000000 00000000 54504C55     ............&<.  ............TPLU
| 2511.00 APPC     31202020 00000000 00000000 E3D7D5C1     ............TPNA  1   ............
| 2511.00 APPC     D4C5F140 40404040 40404040 40404040     ME1              ...@@@@@@@@@@@@
| 2511.00 APPC     40404040 40404040 40404040 40404040                      @@@@@@@@@@@@@@@@
| 2511.00 APPC     40404040 40404040 40404040 40404040                      @@@@@@@@@@@@@@@@
| 2511.00 APPC     40404040 40404040 40404040 00000000                 ....  @@@@@@@@@@@@....
| 2511.00 APPC
| 2511.00 APPC   opcode                          = 14
| 2511.00 APPC   opext                           = 0
| 2511.00 APPC   format                          = 0
| 2511.00 APPC   primary_rc                      = 0  OK
| 2511.00 APPC   secondary_rc                    = 0  OK
| 2511.00 APPC   lu_alias[8]                     = 54504C5531202020
| 2511.00 APPC                                     . & < . . . . .
| 2511.00 APPC                                     T P L U
| 2511.00 APPC   tp_id[8]                        = 0000000000000000
| 2511.00 APPC                                     @ @ @ @ @ @ @ @
| 2511.00 APPC                                     . . . . . . . .
| 2511.00 APPC   tp_name[64]                     = E3D7D5C1D4C5F140
| 2511.00 APPC                                     T P N A M E 1
| 2511.00 APPC                                     . . . @ @ @ @
| 2511.00 APPC   delay_start                     = 0
| 2511.00 APPC   enable_pool                     = 0
| 2511.00 APPC   pip_dlen                        = 0
```

Client/Server Tracing

Client/server tracing records messages that flow between Communications Server for Linux servers in the same domain and between the Communications Server for Linux server and a client. Tracing can be activated on the data flowing between two specific computers or between one computer and all other servers on the LAN. It can be active on either sent or received data or on all data. The following section explains how to perform client/server tracing using the Motif administration program and the command-line administration program.

Performing Client/Server Tracing Using the Motif Administration Program

To perform client/server tracing using the Motif administration program, complete the following steps:

1. Enter **xsnaadmin** from the command line to start the Motif administration program and display the main window.
2. Select the **Diagnostics** menu, then select **Node tracing**. The Tracing dialog is displayed.
3. Select **client-server** to turn on tracing of messages sent between this server, its clients and the other servers in the domain.
4. Optionally, select *Truncate messages* and enter the maximum message size, in bytes. This can be helpful if a large amount of data is being traced and you are interested in the protocol exchanges rather than in the data itself. Protocol data is usually at the start of the individual messages.
5. When you are finished, click **OK**.

Tracing is now enabled and will be written to the following text files:

- **/var/opt/ibm/sna/snacs1.trc**

- **/var/opt/ibm/sna/snacs2.trc**

Performing Client/Server Tracing Using the Command-Line Administration Program

To set up and control client/server tracing using the command-line administration program, complete the following steps:

1. Specify the **snaadmin set_cs_trace** command:
2. Add the *dest_sys* parameter to the **snaadmin** command to specify the client or server name for which tracing is required (this is an ASCII string). Specifying the *dest_sys* parameter enables you to manage tracing on messages flowing between the computer to which this command is issued and one other server on the LAN.

   To manage tracing on messages flowing between the computer to which this command is issued and all other computers on the LAN, do not specify the *dest_sys* parameter.
3. Add the *trace_flags* parameter to the **snaadmin** command to turn all tracing on or off, or to activate tracing on specific message types:
   - To turn all tracing on or off, specify one of the following values:

     ALL Tracing of all types

     NONE No tracing
   - To activate tracing on one or more message types, specify one or more of the following values (use a + character to combine values):

     CS_ADMIN_MSG
     Internal messages relating to client/server topology

     CS_DATAGRAM
     Internal datagram messages

     CS_DATA
     Data messages
4. Add the *trace_direction* parameter with one of the following values to indicate the direction or directions in which tracing is required (this parameter is ignored if *trace_flags* is set to NONE):

   CS_SEND
   Trace messages flowing from the local computer

   CS_RECEIVE
   Trace messages flowing to the local computer

   CS_BOTH
   Trace messages flowing in both directions
5. When you are finished adding parameters and values to the command, press **Enter**.

Tracing is now enabled and will be written to the following text files:

- **/var/opt/ibm/sna/snacs1.trc**
- **/var/opt/ibm/sna/snacs2.trc**

For more information about controlling client/server tracing on Linux computers, refer to the descriptions of the **snaadmin set_cs_trace** and **snaadmin query_cs_trace** commands in the *Communications Server for Linux Administration Command Reference*. (The **snaadmin query_cs_trace** command returns information about the current tracing options for data sent between computers on the

Communications Server for Linux LAN.) For information about controlling client/server tracing on Windows clients, refer to the *Communications Server for Linux Administration Guide*.

Client/Server Trace File Contents

Client/server trace data is written to the following text files (which you can view using a standard ASCII text editor):

/var/opt/ibm/sna/snacs1.trc
Client/server trace file

/var/opt/ibm/sna/snacs2.trc
Backup client/server trace file

The abbreviation **SLM.BS** at the start of each line indicates the SNA LAN Interface Module (SLIM) for Berkeley Software Distribution (BSD) Sockets. This process ID is followed by an indicator of the tracing type (TCP or UDP). Each entry describes an event (such as establishing a connection, or sending or receiving a message) and includes message data where appropriate.

Message data is shown in three columns: hexadecimal, interpreted as EBCDIC, and interpreted as ASCII. Therefore, a text string in the message data appears as readable characters either in the EBCDIC or in the ASCII column according to its character set.

TN Server Tracing

TN server tracing records messages flowing between the Communications Server for Linux TN server and its TN3270 client across the LAN. The following section explains how to perform TN server tracing using the command-line administration program.

Performing TN Server Tracing

To perform TN server tracing using the command-line administration program, complete the following steps:

1. If you do not wish to use the default settings then specify the **snaadmin set_trace_file** command with the *trace_file_type* parameter to specify TN server tracing parameters.
2. Start the TN server tracing by specifying the following administration command. This command must be issued to a running node:

   snaadmin set_tn_server_trace
3. Add the *trace_flags* parameter to the **snaadmin** command to turn all tracing on or off, or to activate tracing on specific message types:
   - To turn all tracing on or off, specify one of the following values:

     ALL Tracing of all types

     NONE No tracing
   - To activate tracing on one or more message types, specify one or more of the following values. Use a + character to combine values:

     TCP TCP/IP interface tracing: messages between the TN server and the TN3270 clients

     FMAPI Node interface tracing: internal control messages, and messages between the TN server and the TN3270 clients (in internal format)

Note: You would not usually need to specify `FMAPI` for first-level diagnostics.

CFG Configuration message tracing: messages relating to the configuration of TN server

4. When you are finished adding parameters and values to the command, press **Enter**.

Tracing is now enabled and will be written to the following text files:

- **/var/opt/ibm/sna/snatnsv1.trc**
- **/var/opt/ibm/sna/snatnsv2.trc**

You can also use the **snaadmin query_tn_server_trace** command to get information about the current tracing options for the Communications Server for Linux TN server feature. For more information about this command and the **snaadmin set_tn_server_trace** command, refer to the *Communications Server for Linux Administration Command Reference*.

TN Server Trace File Contents

TN server trace data is written to **/var/opt/ibm/sna/snatnsv1.trc** and **/var/opt/ibm/sna/snatnsv2.trc**. You can view them using a standard ASCII text editor.

Message data is shown in three columns: hexadecimal, interpreted as EBCDIC, and interpreted as ASCII. Therefore, a text string in the message data appears as readable characters either in the EBCDIC or in the ASCII column according to its character set.

Internal Tracing

Internal tracing traces data flow between Communications Server for Linux processes (the local node and connectivity components). For an illustration of the overall structure of Communications Server for Linux and the types of internal tracing you can specify using the **snaadmin set_trace_type** command, see Figure 1 on page 37. The following section explains how to perform internal tracing using the Motif administration program and the command-line administration program.

Internal tracing is very verbose, and will usually only be used by Communications Server for Linux support personnel.

Controlling Internal Trace Files

You can enable all tracing when SNA starts by issuing **sna start -t**.

To perform internal tracing using the Motif administration program, complete the following steps:

1. Enter **xsnaadmin** from the command line to start the Motif administration program and display the main window.
2. Select the **Diagnostics** menu, then select **Node tracing**. The Tracing dialog is displayed.
3. Select *Set all tracing on* to turn on all internal tracing, or select options in the *Server message trace* section of the dialog to specify tracing on one or more specific areas of Communications Server for Linux (in order to diagnose problems with these areas without collecting large amounts of unrelated trace information). These options allow you to specify the following areas to trace:
   - A particular API or a group of related APIs

- TN Server
- SDLC (this is controlled separately from other trace types of internal tracing, which are controlled using the *Node* option, because it can also be used to provide additional detail in line trace files)
- Node: all types of internal tracing except those for which specific options are displayed in this dialog.

4. Optionally, select *Truncate messages* and enter the maximum message size, in bytes. This can be helpful if a large amount of data is being traced and you are interested in the protocol exchanges rather than in the data itself. Protocol data is usually at the start of the individual messages.
5. When you have done this, click **OK**.

Tracing is now enabled and written to the following binary files:

- **/var/opt/ibm/sna/sna1.trc**
- **/var/opt/ibm/sna/sna2.trc**

To perform internal tracing using the command-line administration program, do the following:

1. Optionally specify the **snaadmin set_trace_file** command with the *trace_file_type* parameter:

   snaadmin set_trace_file, trace_file_type = IPS
2. When you have done this, press **Enter**.
3. Add the *trace_flags* parameter to the **snaadmin set_trace_type** command to specify whether to turn all tracing on or off, or to activate tracing on specific message types:
   - To turn all tracing on or off, specify the *trace_flags* parameter and one of the following values:

     ALL Tracing of all types

     NONE No tracing
   - To activate tracing on one or more of the following message types, specify one or more of the following values. Use a + character to combine values:

     APPC Messages sent between the APPC library and the node.

     FM Messages sent between the 3270 emulation program and the node.

     LUA Messages sent between the LUA library and the node.

     For an SLI application, note that the library converts SLI verbs into the corresponding RUI verbs before sending them to the node. This means that internal tracing for LUA includes only the RUI verbs. Use API tracing to diagnose any problems with the SLI verbs.

     NOF Messages sent between the NOF library and the node.

     MS Messages sent between the MS library and the node.

     NDLC Messages sent between the APPN node and the DLC component.

     LLC2 Messages sent between layers of the LLC2 software.

     MAC Messages sent between layers of the LLC2 software.

     LLI Messages sent over the adapter interface between the LLC2 software and the MAC driver.

     SDLC Messages sent between the SDLC component and the SDLC driver.

As well as producing internal tracing, setting this option also provides additional detail in SDLC line tracing.

NLI Messages sent between the QLLC component and the X.25 driver.

HPRIP Messages sent between the Enterprise Extender (HPR/IP) component and the node.

NODE Messages sent between components within the APPN protocol code.

SLIM Messages sent between master and backup servers in a client/server system.

DGRM Internal control messages sent between system components.

4. When you have finished specifying the syntax, press **Enter**.

For more information about the **snaadmin set_trace_file** and **snaadmin set_trace_type** commands, refer to the *Communications Server for Linux Administration Command Reference*.

Internal Trace File Contents

Internal tracing produces binary trace file output that you can format into text files by using the **snatrcfmt** utility. You can filter the output by using the **snafilter** utility.

For more information about these utilities, see Appendix A, “Using snafilter and snatrcfmt,” on page 53.

Appendix A. Using snafilter and snatrcfmt

Some types of tracing create binary output. Communications Server for Linux provides tools for filtering and formatting these binary files. This appendix describes how to:

- Filter binary trace output to extract only the information you need by using the **snafilter** utility.
- Format binary trace output into text files by using the **snatrcfmt** utility.

Filtering Binary Tracing

The **snafilter** utility enables you to select specific entries from an unformatted, internal trace file so that you can extract only the information you need to diagnose a particular problem. For example, if the trace file contains many different trace types (messages traced at different interfaces within Communications Server for Linux), you can select only messages of a particular type or remove all messages of a particular type. If the trace file contains data from many different APPC or LUA sessions, you can include or exclude messages associated with a particular APPC application or session.

If the trace file contains NLP or RTP frame data from HPR connections, note that snafilter does not filter out these trace types.

Note: You can use snafilter on a line trace file as well as on an internal trace file, or on a single file that contains both types of tracing.

Some of the snafilter options apply only to internal tracing and have no effect on line tracing; this is indicated in the description of each option. Where no mention is made of line tracing, the option applies to both trace types.

The output of snafilter is in a binary format that is suitable for processing by **snatrcfmt**.

Running the snafilter Utility

The syntax for the command to run the trace filter utility is as follows:

snafilter [**-f** *infilename*] [**-o** *outfilename*] [*options*]

Specify the following options and parameters:

-f *infilename*
The input trace file. If you do not use this option, **snafilter** uses **sna1.trc** as the default.

-o *outfilename*
The output trace file. If you do not use this option, **snafilter** uses **snafil.trc** as the default.

+point *tracetype*
Include only messages of the type or types specified (the message types you can specify correspond to the message types described in "Controlling Internal Trace Files" on page 50). Set *tracetype* to **ALL** to turn on tracing of

all types, or specify one or more of the following values. If you want to specify two or more trace types, separate the values with commas, and do not include space characters before or after the commas.

- APPC
- FM
- LUA
- NOF
- MS
- DLC
- LLC2
- MAC
- LLI
- SDLC
- NLI
- HPRIP (for Enterprise Extender links)
- NDLC (node to DLC messages)
- NODE
- SLIM (messages sent between master and backup servers in a client/server system)
- DGRM (Communications Server for Linux internal control messages)

Do not specify both **+point** and **-point**. If you do not specify either option, the default is **+point ALL**.

If the trace file contains both DLC line tracing and internal tracing, you can use **+point DLC** to include only the DLC line tracing.

-point *tracetype*
Exclude messages of the type or types specified. The *tracetype*option is the same as for **+point**, except that **-point ALL** is not valid.

Do not specify both **+point** and **-point**. If you do not specify either option, the default is **+point ALL**.

If the trace file contains both DLC line tracing and internal tracing, you can use **-point DLC** to exclude the DLC line tracing.

+tpid *XXXXXXXXXXXXXXXX*
Include APPC messages with the specified transaction program (TP) ID (in hexadecimal); exclude other APPC messages. This option has no effect on messages other than APPC messages. To specify more than one TP ID, separate them with commas.

Do not specify both **+tpid** and **-tpid**.

This option has no effect on line tracing.

-tpid *XXXXXXXXXXXXXXXX*
Exclude APPC messages with the specified TP ID (in hexadecimal).

Do not specify both **+tpid** and **-tpid**.

This option has no effect on line tracing.

+convid *XXXXXXXX*
Include APPC or CPI-C messages with the specified conversation ID (in hexadecimal); exclude other APPC or CPI-C messages. This option has no

effect on messages other than APPC or CPI-C messages. To specify more than one conversation ID, separate them with commas.

Do not specify both **+convid** and **-convid**.

This option has no effect on line tracing.

-convid *XXXXXXXX*
Exclude APPC or CPI-C messages with the specified conversation ID (in hexadecimal).

Do not specify both **+convid** and **-convid**.

This option has no effect on line tracing.

+sessid *XXXXXXXX*
Include LUA messages with the specified session ID (in hexadecimal); exclude other LUA messages. This option has no effect on messages other than LUA messages. To specify more than one session ID, separate them with commas.

For an SLI application, note that the library converts SLI verbs into the corresponding RUI verbs before sending them to the node. This means that internal tracing for LUA includes only the RUI verbs. Use API tracing to diagnose any problems with the SLI verbs.

Do not specify both **+sessid** and **-sessid**.

This option has no effect on line tracing.

-sessid *XXXXXXXX*
Exclude LUA messages with the specified session ID (in hexadecimal).

Do not specify both **+sessid** and **-sessid**.

This option has no effect on line tracing.

+lfsid *aabbc*
Include DLC messages with the specified local-form session identifier (LFSID); exclude other DLC messages. This option has no effect on messages other than DLC messages.

The LFSID consists of a 2-character OAF (*aa*) in hexadecimal, a 2-character DAF (*bb*) in hexadecimal, and a one-character ODAI (*c*), formatted in the same way as they are used in messages flowing from the local node. You can use the wildcard characters *xx* instead of *aa* or *bb*, and you can use *x* instead of *c*. To specify more than one LFSID, separate them with commas.

This option controls messages in both directions on the session; you cannot filter on messages in a single direction only. For example, if messages flowing from the node have OAF = 01, DAF = 02, and ODAI = 1, messages flowing to the node on the same session will have OAF = 02, DAF = 01, and ODAI = 1. Specifying **+lfsid 01021** includes messages flowing in both directions on this session.

You can use both of the options **+lfsid** and **-lfsid**.

-lfsid *aabbc*
Exclude DLC messages with the specified LFSID.

You can use both of the options **+lfsid** and **-lfsid**. For example, to include all messages with DAF `0x0a`except those with OAF `0x0b`, specify **+lfsid xx0ax -lfsid 0bxxx**.

+npid *XXXXXXXX*
Include node and SDLC messages with the specified component instance ID (in hexadecimal); exclude other nodeand SDLC messages. This option has no effect on messages other than node and SDLC messages.

The component instance ID is a Communications Server for Linux internal identifier that distinguishes between different users or programs using the same component. To specify more than one component instance ID, separate them with commas.

Do not specify both **+npid** and **-npid**.

This option has no effect on line tracing.

-npid *XXXXXXXX*
Exclude node and SDLC messages with the specified component instance ID (in hexadecimal).

Do not specify both **+npid** and **-npid**.

This option has no effect on line tracing.

-start *yymmddhhmmss*
Include only messages traced after the specified date (year, month, day) and time (hours, minutes, seconds). For example, 3:45 p.m. (15.45) on 11 August 1997 is **970811154500**.

You can specify both **-start** and **-end** to include only messages between the specified times. The end time you specify must be later than the start time.

-end *yymmddhhmmss*
Exclude messages traced after the specified date and time. Use the same format as **-start**.

You can specify both **-start** and **-end** to include only messages between the specified times. The end time you specify must be later than the start time.

Sample Command and Output

The following sample trace command illustrates some of the command options:

snafilter -f new.trc -o newout.trc +point APPC,NOF,DLC -lfsid 0a021 +convid 0100000a

This command does the following:

- Takes input from the file **new.trc**
- Sends the output to the file **newout.trc**
- Includes only APPC, NOF, and DLC messages
- Excludes DLC messages with LFSID 0a021
- Includes only APPC messages with conversation ID 0100000a

The output file then contains the following:

- All APPC messages with the specified conversation ID
- All NOF messages
- All DLC messages except those with the specified LFSID

You can then format this filtered output by using the **snatrcfmt** command, which is explained in the next section.

Formatting Internal Binary Trace Output into Text Files

The **snatrcfmt** command-line utility enables you to format binary trace output into one or two text files. It can be used either for line tracing or for internal tracing, with slightly different options in each case.

- For line tracing, you can format the binary trace output into only one text file, a message data dump file.
- For internal tracing, you can format the binary trace output into either a message data dump file (see "Message Data Dump" on page 62) or a message flow diagram (see "Message Flow Drawing" on page 60), or both. The message flow diagram is a summary drawing that shows the message flows between components.
- If your binary trace file contains both line tracing and internal tracing, both trace types are included in the output file.

Running the snatrcfmt Utility for Line Tracing

The syntax of the command to run the trace format utility for line tracing is:

snatrcfmt [**-f** *filename*] [**-o** *output_file_base*] [
-S | **-D**] [**-m**] [
-l] [**-M**]

Specify the following options and parameters:

-f *filename*
Use this option to specify the name of the input binary trace file. If you do not use this option, **snatrcfmt** uses **sna1.trc** as the default.

-o *output_file_base*
The base name to be used for the output file. Communications Server for Linux adds the extension **.dmp** for the message data dump file. If you do not use this option, Communications Server for Linux uses the files **snatrc.dmp** as the output file.

-S Produce a summary trace file, with only one line of tracing for each message. If the trace file includes detailed SDLC tracing, this suppresses Information fields and includes only address and control fields.

-D Produce a more detailed report of each message.

-m For each message, show the time interval since the last message instead of the absolute time. If you do not select this option, each message shows the time and date at which it was written to the file.

-l Suppress detailed SDLC tracing if it is included in the trace file (but include standard SDLC line trace). If you do not select this option, any detailed SDLC tracing from the trace file is included in the output.

-M If detailed SDLC tracing is included in the file, decode the frames in Modulo 128 format. If you do not select this option, detailed SDLC tracing is decoded in Modulo 8 format.

Running the snatrcfmt Utility for Internal Tracing

The syntax of the command to run the trace format utility for internal tracing is:

snatrcfmt -i [**-m**] [
-f *filename*] [**-o** *output_file_base*] [*options*]

Specify the following options and parameters:

-i Use this option to indicate that snatrcfmt is being used to format internal tracing.

-m For each message, show the time interval since the last message instead of the absolute time. If you do not select this option, each message shows the time and date at which it was written to the file.

-f *filename*
Use this option to specify the name of the input binary trace file. If you do not use this option, **snatrcfmt** uses **sna1.trc** as the default.

-o *output_file_base*
The base name to be used for the output files. Communications Server for Linux adds the extension **.drw** for the message flow drawing and **.dmp** for the message data dump. If you do not use this option, Communications Server for Linux uses the files **snatrc.drw** and **snatrc.dmp** as the output files.

The options indicated above by *options* are as follows:

-w Only create the message data dump file.

The following options are used only for the message data dump file. Do not specify any of these options with the **-W** option.

-b Include a listing of each message as it is sent by one component and received by another. If you do not specify this option, Communications Server for Linux includes only sent messages.

-r Include only raw hexadecimal data for each message; do not attempt to interpret the message data.

-d Include verbose formatting for each message. Communications Server for Linux lists the data in three columns: hexadecimal, interpreted as EBCDIC, and interpreted as ASCII (so that a text string in the message data appears as readable characters either in the EBCDIC or in the ASCII column, according to its character set). In addition, Communications Server for Linux interprets many of the fields in the message data and prints out the interpretation as readable text.

-D As for **-d**, but with a detailed listing of the fields in the RH and TH of each message.

Do not specify more than one of the options **-r**, **-d**, and **-D**. If you do not specify any of these options, Communications Server for Linux includes the hexadecimal, EBCDIC, and ASCII listing but interprets only a limited number of message fields.

-W Only create the message flow drawing.

The following options are used only for the message flow drawing file. Do not specify any of the following options with the **-w** option.

-c *component_group*
Compress a group of components into one column so that messages between these components do not appear in the drawing. Specify one or more of the following separated by commas:

CP Compress control point into one column.

LU Compress LU 6.2 components (CPI-C and APPC) into one column.

OL Compress LU 0–3 components into one column.

NODE Compress all components in the APPN protocol code [including the CP, LU, and Old LU (OL) groups] into one column so that internal messages within the node are not shown.

NONE No compression. Do not specify any other component options if you select this option.

USER Compress all API components into one column.

SIX Compress all internal components (except DLCs) outside the APPN protocol code into one column.

DLC Compress all DLC components into one column.

If you do not use this option, the default is **-c** `USER,NODE`. If you use both of the options **-c** and **-s**, you must specify **-c** before **-s**.

-s *components*
Do not display a column in the drawing for one or more individual components. Any messages flowing to or from these components are shown in the "unknown" column, so you can view detailed information without having to display all the components in a group. For example, you can view the PS and HS components but suppress the RM component.

Specify each component by using the two-character component identifiers as listed in "Message Flow Drawing" on page 60. To specify multiple components, separate them with commas.

Note: Note that you specify component groups with the **-c** option, but individual components with the **-s** option.

If you use both of the options **-c** and **-s**, you must specify **-c** before **-s**.

-p *nn* Use a page length of *nn* lines for the message flow drawing (*nn* is a decimal number). Communications Server for Linux separates pages with a page break character and includes the column headers at the top of each page.

Do not specify both of the options **-p** and **-P**. If you do not use either option, the default is **-p** `66`.

-P Produce output as a single page (no page breaks and column headers only at the start of the data).

Do not specify both of the options **-p** and **-P**. If you do not use either option, the default is **-p** `66`.

Output from the snatrcfmt Utility

The **snatrcfmt** utility generates text output in either or both of the following formats:

Message flow drawing
A drawing showing the messages flowing between different Communications Server for Linux components (this format does not apply to line tracing).

Message data dump
A listing of the data in each trace message.

Message Flow Drawing

Each column in the message flow drawing corresponds to a particular Communications Server for Linux component or to a group of components. The heading of each column shows a one-character or two-character abbreviation for the name of the component or group. Each line in the file corresponds to a particular message flowing between Communications Server for Linux components.

The default options group all API components into one column, and all components in the APPN protocol code into another column. The only messages shown are those flowing between API components and the node, and those flowing between the node and DLC components. The drawing also includes an individual column for each of the DLC components. If required, you can break these groups down into individual components to show a more detailed drawing; the additional detail is provided mainly for use by Communications Server for Linux support personnel.

Note: Formatted output varies depending upon the options you select. Because support personnel sometimes use different options, always provide them with the original binary trace files.

Node line tracing is not shown in message flow drawings because the data being traced is flowing between Communications Server for Linux and a remote component (which is not shown in the diagram) rather than between two Communications Server for Linux components.

The component groups shown in the message flow drawing are as follows:

US The following API components:

- **AL** APPC library
- **CL** CPI-C library
- **CV** CSV library
- **RL** RUI (LUA) library

  For an SLI application, note that the library converts SLI verbs into the corresponding RUI verbs before sending them to the node. This means that internal tracing for LUA includes only the RUI verbs. Use API tracing to diagnose any problems with the SLI verbs.
- **ML** MS library
- **NL** NOF library
- **CD** Config daemon
- **LD** Log daemon
- **SD** SNA daemon
- **RD** RCF daemon
- **RS** Remote system (client/server messages)

SIX The following internal components (except DLCs) outside APPN protocol code:

- **PM** Path manager

SV Service manager

AP, TP
APPC internal components

LO Log internal component

M MS internal component

N NOF internal component

L1, L2 LUA internal components

FM FM internal component

CP Control point, which consists of:

- **CM** Session connector manager
- **NF** NOF node component
- **AM, AS**
  Address space manager
- **CS** Configuration services
- **DS** Directory services
- **MD** Multiple Domain Support (MDS) component of management services
- **MS** Management services
- **SS** Session services
- **TR** Topology and routing services
- **DR, ES**
  Dependent LU Requester (DLUR)

LU LU 6.2, which consists of:

- **PS** Presentation services
- **HS** Half-session
- **RM** Resources manager
- **SM** Session manager

OL Old LU (LU 0–3), which consists of:

- **RU** RUI (LUA)
- **CH** Conventional half-session
- **LM** LU manager
- **PU** PU manager
- **PX** SNA gateway

NO The following node components (components of APPN protocol code):

- **BM** Buffer manager
- **D** DLC component within a node
- **PC** Path control
- **SC** Session connector

HP, RT
High Performance Routing

DL DLC components, which consist of:

L, LL, MT, M1
LLC2 driver

SL SDLC driver

S2 SDLC device driver

HM SDLC hardware interface

QL QLLC driver

X2 X.25 interface (NLI)

IP Enterprise Extender (HPR/IP)

In addition to grouping components together, you can also suppress the column for a particular component or group. Any messages flowing to or from this component are shown as flowing to or from the "unknown" column, which is marked **??**.

Each line in the drawing ends with a number prefixed with a **$** character, followed by a timestamp. The number indicates the line number at which the message is listed in the corresponding message data dump file, and the timestamp shows the time the message was generated.

The following example shows the format of the message flow drawing:

```
Message Flow Drawing Example
File: sna1.trc     Page 16     Trace started: Tue Apr  4 10:56:41.250 GMT 2000
+--++--++--------------++--++--------------++--+
|US||CP||PS  HS  RM  SM||OL||SC  PC  D   BM||GG|
+--++--++--------------++--++--------------++--+
.   .ASSIGN_LFSID   |   .   .   .   .   .   .      $013795 10:45:48.120
.   o<--------------+   .   .   .   .   .   .      $013795 10:45:48.120
.   |ASSIGN_LFSID_RSP   .   .   .   .   .   .      $013815 10:45:48.120
.   +-------------->o   .   .   .   .   .   .      $013815 10:45:48.120
.   .   .   .CREATE |   .   .   .   .   .   .      $013835 10:45:48.120
.   .   .   o<------+   .   .   .   .   .   .      $013835 10:45:48.120
.   .MU(MU_BIND_RQ_SEND)RQD1.   .   .   .   .      $013845 10:45:48.120
.   o<--------------+   .   .   .   .   .   .      $013845 10:45:48.120
.   |MU(MU_BIND_RQ_SEND)RQD1,PI .   .   .   .      $013900 10:45:48.310
.   +------------------------->o   .   .   .      $013900 10:45:48.310
.   .   .   .   .   .   .   .   |DLC_MU .   .      $014010 10:45:48.310
.   .   .   .   .   .   .   .   +-->o   .   .      $014010 10:45:48.310
.   .   .   .   .   .   .   .   .   |DLC_MU .      $014065 10:45:48.310
.   .   .   .   .   .   .   .   .   +------>o      $014065 10:45:48.310
```

Message Data Dump

The first few lines of the message data dump file contain identification and field alignment information about the running system. This information is used only by Communications Server for Linux support personnel.

For each message, the file includes header information about the source, destination, and type of the message, followed by a hexadecimal listing of the message data. You have the choice of three levels of detail for the message data:

- Uninterpreted hexadecimal data
- Hexadecimal data interpreted as EBCDIC and as ASCII

- Hexadecimal data interpreted as EBCDIC and as ASCII, with text interpretations of many of the message fields

If the message data dump is from a trace file containing line trace and including detailed SDLC tracing, the following additional information is included for an SDLC frame:

- "TX" or "RX", indicating whether the frame is being transmitted or received by Communications Server for Linux
- Decoded versions of the address and control fields (and Information fields if they are present).

The following examples show the format of the message data dump for line tracing:

```
Message Data Dump Example for Line Tracing
------------------------------------------------ 16:54:33.950 BST   20 Apr 1999
SND>> CNCT_OUT REQ                                  ETHER0.ETSAP0.ETHL0
------------------------------------------------ 16:54:33.950 BST   20 Apr 1999
SND>> CNCT_OUT REQ                                  TOKEN0.TRSAP0.TRL0
------------------------------------------------ 16:54:33.950 BST   20 Apr 1999
<<RCV CNCT_OUT RSP OK                               ETHER0.ETSAP0.ETHL0
------------------------------------------------ 16:54:33.950 BST   20 Apr 1999
SND>> XID (NULL)                                    ETHER0.ETSAP0.ETHL0
------------------------------------------------ 16:54:33.960 BST   20 Apr 1999
<<RCV CNCT_OUT RSP OK                               TOKEN0.TRSAP0.TRL0
------------------------------------------------ 16:54:33.960 BST   20 Apr 1999
SND>> XID (NULL)                                    TOKEN0.TRSAP0.TRL0
------------------------------------------------ 16:54:33.960 BST   20 Apr 1999
<<RCV XID (NULL)                                    ETHER0.ETSAP0.ETHL0
------------------------------------------------ 16:54:33.960 BST   20 Apr 1999
SND>> XID FMT:3 ID:01100002 ESI:PRE_NEG LR:SEC      ETHER0.ETSAP0.ETHL0
XID   32540110 00020000 000AD100 00000000   .........J..... 2T.............
      00010B41 00040900 00000007 000E0AF4   ................4 ...A...........
      C1D7D7D5 4BD4D6D6 D5102900 28110C08   APPN.MOON....... ....K.....).(...
      04F0F6F0 F0F0F609 06E2D5C1 E2E3C1D9   .060006..SNASTAR .............`..
      03084011 0FE2D5C1 E2E3C1D9 40D3C9D4   .. ..SNASTAR LIM ..@.........@...
      C9E3C5C4                              ITED             ....
------------------------------------------------ 16:54:33.970 BST   20 Apr 1999
<<RCV XID (NULL)                                    TOKEN0.TRSAP0.TRL0
Message Data Dump Example for Line Tracing with Maximum Detail
------------------------------------------------ 16:54:33.950 BST   20 Apr 1999
SND>> CNCT_OUT REQ                                  ETHER0.ETSAP0.ETHL0
IPS:  00000000 4554484C 30202020 010000EE   .......<........ ....ETHL0   ....
      01000008 00000007 000629EA BC670400   ................ ..........)..g..
      00000000 00000000 00000000 00000000   ................ ................
      00000000 00000000 040900FF 8100009C   ............@a... ................
      FFFFFFFF 00020040 00060000 00000000   @@@@... ........ .......@........
      00000000 00000000 00000000 00000000   ................ ................
      00000000 00000000 00000000 00000000   ................ ................
      0000000A 0005000A 00051388 13880064   ............h.h.. ...............d
      000A001E 001E0003                     ........         ........
------------------------------------------------ 16:54:33.950 BST   20 Apr 1999
SND>> CNCT_OUT REQ                                  TOKEN0.TRSAP0.TRL0
IPS:  00000000 54524C30 20202020 010000F0   ......<........0 ....TRL0    ....
      50182F10 00000007 08005AFD 90B30400   &.........!..... P./.......Z.....
      00000000 00000000 00000000 00000000   ................ ................
      00000000 00000000 1009000D 810000A5   ............a..v ................
      000E0007 00020040 00050000 00000000   ....... ........ .......@........
      00000000 00000000 00000000 00000000   ................ ................
      00000000 00000000 00000000 00000000   ................ ................
      0000000A 0005000A 00051388 13880064   ............h.h.. ...............d
      000A001E 001E0003                     ........         ........
------------------------------------------------ 16:54:33.950 BST   20 Apr 1999
<<RCV CNCT_OUT RSP OK                               ETHER0.ETSAP0.ETHL0
```

```
IPS:  00000001 4554484C 30202020 010000EE   .......<........ ....ETHL0   ....
      410000EF 00000007 000629EA BC670400   ................ A.........)..g..
      00000000 00000000 00000000 00000000   ................ ................
      00000000 00000000 040900FF 0100011E   ...........@.... ................
      FFFFFFFF 00020040 00060000 00000000   @@@@... ........ .......@........
      00000000 00000000 00000000 00000000   ................ ................
      00000000 00000000 00000000 00000000   ................ ................
      0000000A 0005000A 00051388 13880064   ...........h.h.. ...............d
      000A001E 001E0003                     ........         ........
------------------------------------------------ 16:54:33.950 BST   20 Apr 1999
SND>> XID (NULL)                                    ETHER0.ETSAP0.ETHL0
IPS:  410000EF 00020000                     ........         A.......
------------------------------------------------ 16:54:33.960 BST   20 Apr 1999
<<RCV CNCT_OUT RSP OK                               TOKEN0.TRSAP0.TRL0
IPS:  00000001 54524C30 20202020 010000F0   ......<........0 ....TRL0    ....
      410000F1 00000007 08005AFD 90B30400   ...1......!..... A.........Z.....
      00000000 00000000 00000000 00000000   ................ ................
      00000000 00000000 1009000D 0100012D   ................ ................-
      000E0007 00020040 00050000 00000000   ....... ........ .......@........
      00000000 00000000 00000000 00000000   ................ ................
      00000000 00000000 00000000 00000000   ................ ................
      0000000A 0005000A 00051388 13880064   ...........h.h.. ...............d
      000A001E 001E0003                     ........         ........
------------------------------------------------ 16:54:33.960 BST   20 Apr 1999
SND>> XID (NULL)                                    TOKEN0.TRSAP0.TRL0
IPS:  410000F1 00020019                     ...1....         A.......
------------------------------------------------ 16:54:33.960 BST   20 Apr 1999
<<RCV XID (NULL)                                    ETHER0.ETSAP0.ETHL0
IPS:  010000EE 80370070                     ........         .....7.p
------------------------------------------------ 16:54:33.960 BST   20 Apr 1999
SND>> XID FMT:3 ID:01100002 ESI:PRE_NEG LR:SEC      ETHER0.ETSAP0.ETHL0
IPS:  410000EF 00025400                     ........         A.....T.
XID decode:

XID format type                 = 03
Node type                       = 02
XID length                      = 54
block/ID number                 = 01100002
                                  . . @ .
                                  . . . .
Bytes 8-9                       = 000A
            Init self may be sent to the XID sender
            XID sender supports independent-PLU session partners
            This node can generate BIND PIU segments
            This node can receive BIND PIU segments
            ACTPU for an SSCP-PU session requested
            The XID sender is not a network node
            CP services not requested or supported
            CP-CP sessions not supported on this TG
            Secondary initiated non-activation exchange supported
            XID sender does not supported CP name change
            Prenegotiation exchange
Byte 10                         = D1
            Adaptive BIND pacing as a BIND sender supported
            Adaptive BIND pacing as a BIND receiver supported
            This TG is operative
            XID sender supports receipt of ACTPU containing PU cap cv
            XID sender is not a peripheral border node
            Adaptive pacing on both, negotiable
Byte 15                         = 00
            XID sender does NOT support parallel TGs
TG number                       = 00
DLC type                        = 01
DLC type is SDLC
DLC data length                 = 0B
Byte 19                         = 41
            XID sender can be an ABM combined station
```

```
            XID sender not already using short-hold mode
            Short-hold mode not supported
            Sender is secondary link station (non-negotatiable)
            Link-station transmit-receive capability: two-way simultaneous
Byte 20                         = 00
            XID sender is not the sender of a nonactivation XID
Maximum BTU length              = 409
Byte 23                         = 00
            SNA link profile
Byte 24                         = 00
            SIM and RIM not supported
I-frame number                  = 07
XID   32540110 00020000 000AD100 00000000   .........J..... 2T.............
      00010B41 00040900 00000007 000E0AF4   ..............4 ...A............
      C1D7D7D5 4BD4D6D6 D5102900 28110C08   APPN.MOON....... ....K.....).(...
      04F0F6F0 F0F0F609 06E2D5C1 E2E3C1D9   .060006..SNASTAR .............`..
      03084011 0FE2D5C1 E2E3C1D9 40D3C9D4   .. ..SNASTAR LIM ..@.........@...
      C9E3C5C4                              ITED             ....
------------------------------------------------ 16:54:33.970 BST   20 Apr 1999
<<RCV XID (NULL)                                    TOKEN0.TRSAP0.TRL0
IPS:  010000F0 5A000000                     ...0!...         ....Z...
```

Appendix B. Using getsense

SNA network failures are indicated by sense codes that are returned to application programs. The SNA sense codes appear in internal service logs as eight-digit hexadecimal values (four bytes):

- The first two digits indicate the failure category.
- The next two digits indicate the failure category modifier.
- The last four digits indicate the failure subcategory. The failure subcategory gives detailed, specific information about the nature of the failure.

To retrieve information about a specific SNA sense code generated on the Communications Server for Linux computer, type **sna -getsense** followed by either the category and modifier (the first four digits) or the entire sense code (all eight digits) on the command line.

For example, to obtain information about sense code 08170001, type the following:

sna -getsense 08170001

You can use **sna -g** as a shortened form of the command **sna -getsense**.

The output from the command is as follows:

```
# sna -getsense 08170001
  REQUEST REJECT (CATEGORY CODE = X'08')
  ______________________________________

  This category indicates that the request was delivered to
  the intended component and was understood and supported,
  but not executed.

  0817      Link or Link Resource Inactive:  A request
            requires the use of a link or link resource
            that is not active.

      0001  Link inactive.
```

If the **sna -getsense** command does not recognize the sense code specified, it attempts to retrieve the failure category and failure category modifier information (the first four digits). If the **sna -getsense** command is unable to retrieve this information, refer to the *IBM Systems Network Architecture: Formats* book.

If the SNA sense code was generated on a remote computer, you may need to use an equivalent of **sna -getsense** on that computer to determine its meaning.

Appendix C. Using snagetpd

You occasionally may need to send files to support personnel so that they can diagnose problems. The diagnostic collection utility, **snagetpd**, is a command-line program that enables you to easily gather the information required for support personnel into a single file.

snagetpd collects

- Information about current settings of the log and trace utilities, such as whether audit logging was active and the size of the log files
- Log and trace files
- Core files (if required)
- Contents of the diagnostics file directory **/var/opt/ibm/sna**
- On a Remote API Client on AIX or Linux: the client network data file **sna_clnt.net**, and the file **server.current** that records details of the server to which the client is currently connected

This appendix explains how to use **snagetpd**.

Operating snagetpd

If you encounter a problem running Communications Server for Linux that you cannot resolve, support personnel might ask you to run the **snagetpd** utility and send them the output file containing the diagnostic data.

The **snagetpd** utility must be run from a root login. It produces a compressed tar file containing many files, including output files from the Communications Server for Linux tracing and logging utilities.

Before it starts to collect files, **snagetpd** deactivates all types of Communications Server for Linux tracing that are controlled by the administration tools (such as line tracing, Client/Server tracing, TN Server tracing, and internal tracing). It does not change the settings for logging, or for user-space API tracing (controlled by environment variables).

During program execution, **snagetpd** prompts you to describe your problem by displaying the message, **Please describe the symptoms of the problem**. After entering the problem description, either press **CTRL+D** or type **$** and press **Enter** to continue running the program.

When the program has completed, send the output file (see "Command Syntax and Program Output" on page 70) to your support staff for diagnosis.

Note:

1. The **snagetpd** utility renames some of the diagnostics files within the tar file, so you may find that the contents do not match your original filenames. The utility does not change or rename the original diagnostics files on your system.

2. If the problems occur on a Linux client computer, run **snagetpd** on both the client computer and the server containing the resources it is using, to ensure that you obtain as much problem determination information as possible.

Command Syntax and Program Output

The command syntax for **snagetpd** is as follows:

snagetpd [**-q**] [*filename*]

Specify the following options and parameters:

-q Specifies quiet mode. When this option is specified, **snagetpd** runs without prompting you for information. In addition to collecting the log and trace files, and information about current settings of the log and trace utilities, **snagetpd** collects any core files called **core*** that are in the local directory.

If you do not specify **-q** then **snagetpd** asks you for a description of your problem and prompts you for the path for any core files that you wish to collect and reminds you to collect the associated executable files.

filename

Specifies the name of the output file that contains the problem determination information. This output file is placed in the current directory unless you specify a path when you use the *filename* parameter.

If you specify a file name, the output of **snagetpd** is *filename*.tar.gz. If you do not specify a file name, Communications Server for Linux assigns the default file name, **pd.tar.gz**.

To place an output file called **snaperr.tar.gz** in the **pd** subdirectory, enter the following on the command line:

snagetpd `pd/snaperr`

Command Restrictions

The following restrictions apply to **snagetpd**:

- If you run **snagetpd** on a machine that does not have sufficient disk space to store the entire contents of the output file, **snagetpd** displays the error message **Insufficient disk space**. If this occurs, the output is a tarred file consisting of those files that were collected before running out of disk space. The file is named *filename* **.tar**.
- If central logging is active and you run **snagetpd** on a node other than the node acting as the central logger, any logging information collected on the central logger is not included in the output file. However, any information accessible to the node on which you run **snagetpd** is included in the output file.
- If you run **snagetpd** when Communications Server for Linux is not running, the following data is not included in the output file:
  - Information about the settings of the log and trace utilities when **snagetpd** was run, such as whether audit logging was active and the size of the log files.
  - Any log and trace files not located in the default directories with the default file names. (For example, **snagetpd** searches for tracing information in the **/var/opt/ibm/sna** subdirectory for files named * **.trc**.)

Appendix D. Windows Clients

This appendix describes logging and tracing information that is specific to Windows clients.

Logging for Windows Clients

Logging for Windows clients is controlled by options in the Registry, as described in the *Communications Server for Linux Administration Guide*. Problem and exception messages are logged to the error log file, and audit messages are logged to the audit log file.

Problem messages are always logged and cannot be disabled, but you can specify whether to log exception and audit messages. If either of these options is not explicitly specified, issue the **snaadmin set_global_log_type** command on the server:

If central logging is enabled (by issuing the **snaadmin set_central_logging** command on the server), all messages from the client are written to the central log file. Otherwise, you can specify the following:

- Local files on the Windows client to hold error and audit messages
- Files to be used for backing up log information
- The size at which log files are backed up and reset
- Whether to use verbose logging or succinct logging

Log files are backed up and reset in the same way as they are for Linux computers, except that the default maximum size for a Windows log file is 10,000 bytes (not 1,000,000 bytes as for Linux).

Controlling Tracing on Windows Clients

On a Windows client, Communications Server for Linux provides facilities for API tracing and client/server tracing. Options in the Registry control all of these trace types. For more information, refer to the *Communications Server for Linux Administration Guide*.

The format of each of these trace types is the same as for components on Linux computers (for more information, see "Client/Server Tracing" on page 47, "Trace File Format for API Tracing" on page 43).

Communications Server for Linux does not provide tracing facilities for 5250 emulation programs or for HLLAPI applications on Windows clients. However, because 5250 data is transferred using APPC, you can use APPC API tracing on the client to trace the data sent from a 5250 emulation program to the node (for more information, refer to the *Communications Server for Linux Administration Guide*). Additional tracing facilities may be provided with the 5250 or 3270 emulation software; refer to the documentation supplied with the program for more information.

Collecting Diagnostic Information on Windows Clients

You occasionally may need to send diagnostic files to support personnel so that they can diagnose problems. The diagnostic collection utility, **snagetpd**, is a command-line program that enables you to easily gather the information required for support personnel into a single file.

To run **snagetpd** on a Windows client, type the following command in a command window or from the Start / Run icon:

snagetpd

The **snagetpd** utility collects together all of the Windows trace files and other Windows Client system information, and gathers them into a single self-extracting ZIP file named **snapd.exe**, which you can send to support personnel.

Appendix E. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
P.O. Box 12195
3039 Cornwallis Road
Research Triangle Park, NC 27709-2195
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE: This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows: ® (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. ® Copyright International Business Machines Corporation. 1998, 2007. All rights reserved.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

Advanced Peer-to-Peer Networking®
AIX
Application System/400®
APPN
AS/400®
CICS®
DB2®
Enterprise System/3090™
Enterprise System/4381™
Enterprise System/9000®
ES/3090™
ES/9000®
eServer™
IBM
IBMLink™
IMS™
MVS™
MVS/ESA™
NetView
Operating System/2®
Operating System/400®
OS/2®
OS/400®
PowerPC®
PowerPC Architecture™
pSeries
S/390®
System/390®
System p5™
System z
System z9™
System 390
VSE/ESA™
VTAM®
WebSphere®
z/OS
z/9

The following terms are trademarks or registered trademarks of other companies:

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc., in the United States, other countries, or both.

UNIX® is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

Intel and EM64T are trademarks of Intel Corporation.

AMD64 is a trademark of Advanced Micro Devices, Inc.

Linux is a trademark of Linus Torvalds.

RedHat and RPM are trademarks of Red Hat, Inc.

SuSE Linux is a trademark of Novell.

Microsoft®, Windows, Windows 2003, Windows XP, Windoes Vista, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

Bibliography

The following IBM publications provide information about the topics discussed in this library. The publications are divided into the following broad topic areas:

- Communications Server for Linux, Version 6.2.3
- Systems Network Architecture (SNA)
- Host configuration
- z/OS® Communications Server
- Transmission Control Protocol/Internet Protocol (TCP/IP)
- X.25
- Advanced Program-to-Program Communication (APPC)
- Programming
- Other IBM networking topics

For books in the Communications Server for Linux library, brief descriptions are provided. For other books, only the titles, order numbers, and, in some cases, the abbreviated title used in the text of this book are shown here.

Communications Server for Linux Version 6.2.3Publications

The Communications Server for Linux library comprises the following books. In addition, softcopy versions of these documents are provided on the CD-ROM. See *IBM Communications Server for Linux Quick Beginnings* for information about accessing the softcopy files on the CD-ROM. To install these softcopy books on your system, you require 9–15 MB of hard disk space (depending on which national language versions you install).

- *IBM Communications Server for Linux Quick Beginnings* ()

  This book is a general introduction to Communications Server for Linux, including information about supported network characteristics, installation, configuration, and operation.

- *IBM Communications Server for Linux Administration Guide* ()

  This book provides an SNA and Communications Server for Linux overview and information about Communications Server for Linux configuration and operation.

- *IBM Communications Server for Linux Administration Command Reference* ()

  This book provides information about SNA and Communications Server for Linux commands.

- *IBM Communications Server for Linux CPI-C Programmer's Guide* ()

  This book provides information for experienced "C" or Java programmers about writing SNA transaction programs using the Communications Server for Linux CPI Communications API.

- *IBM Communications Server for Linux APPC Programmer's Guide* ()

  This book contains the information you need to write application programs using Advanced Program-to-Program Communication (APPC).

- *IBM Communications Server for Linux LUA Programmer's Guide* ()

  This book contains the information you need to write applications using the Conventional LU Application Programming Interface (LUA).

- *IBM Communications Server for Linux CSV Programmer's Guide* ()

  This book contains the information you need to write application programs using the Common Service Verbs (CSV) application program interface (API).
- *IBM Communications Server for Linux MS Programmer's Guide* ()

  This book contains the information you need to write applications using the Management Services (MS) API.
- *IBM Communications Server for Linux NOF Programmer's Guide* ()

  This book contains the information you need to write applications using the Node Operator Facility (NOF) API.
- *IBM Communications Server for Linux Diagnostics Guide* ()

  This book provides information about SNA network problem resolution.
- *IBM Communications Server for Linux APPC Application Suite User's Guide* ()

  This book provides information about APPC applications used with Communications Server for Linux.
- *IBM IBM Communications Server for Linux Glossary* ()

  This book provides a comprehensive list of terms and definitions used throughout the IBM IBM Communications Server for Linux library.

Systems Network Architecture (SNA) Publications

The following books contain information about SNA networks:

- *Systems Network Architecture: Format and Protocol Reference Manual—Architecture Logic for LU Type 6.2* (SC30-3269)
- *Systems Network Architecture: Formats* (GA27-3136)
- *Systems Network Architecture: Guide to SNA Publications* (GC30-3438)
- *Systems Network Architecture: Network Product Formats* (LY43-0081)
- *Systems Network Architecture: Technical Overview* (GC30-3073)
- *Systems Network Architecture: APPN Architecture Reference* (SC30-3422)
- *Systems Network Architecture: Sessions between Logical Units* (GC20-1868)
- *Systems Network Architecture: LU 6.2 Reference—Peer Protocols* (SC31-6808)
- *Systems Network Architecture: Transaction Programmer's Reference Manual for LU Type 6.2* (GC30-3084)
- *Systems Network Architecture: 3270 Datastream Programmer's Reference* (GA23-0059)
- *Networking Blueprint Executive Overview* (GC31-7057)
- *Systems Network Architecture: Management Services Reference* (SC30-3346)

Host Configuration Publications

The following books contain information about host configuration:

- *ES/9000, ES/3090 IOCP User's Guide Volume A04* (GC38-0097)
- *3174 Establishment Controller Installation Guide* (GG24-3061)
- *3270 Information Display System 3174 Establishment Controller: Planning Guide* (GA27-3918)
- *OS/390 Hardware Configuration Definition (HCD) User's Guide* (SC28-1848)

z/OS Communications Server Publications

The following books contain information about z/OS Communications Server:

- *z/OS V1R7 Communications Server: SNA Network Implementation Guide* (SC31-8777)

- *z/OS V1R7 Communications Server: SNA Diagnostics* (Vol 1: GC31-6850, Vol 2: GC31-6851)
- *z/OS V1R6 Communications Server: Resource Definition Reference* (SC31-8778)

TCP/IP Publications

The following books contain information about the Transmission Control Protocol/Internet Protocol (TCP/IP) network protocol:

- *z/OS V1R7 Communications Server: IP Configuration Guide* (SC31-8775)
- *z/OS V1R7 Communications Server: IP Configuration Reference* (SC31-8776)
- *z/VM V5R1 TCP/IP Planning and Customization* (SC24-6125)

X.25 Publications

The following books contain information about the X.25 network protocol:

- *Communications Server for OS/2 Version 4 X.25 Programming* (SC31-8150)

APPC Publications

The following books contain information about Advanced Program-to-Program Communication (APPC):

- *APPC Application Suite V1 User's Guide* (SC31-6532)
- *APPC Application Suite V1 Administration* (SC31-6533)
- *APPC Application Suite V1 Programming* (SC31-6534)
- *APPC Application Suite V1 Online Product Library* (SK2T-2680)
- *APPC Application Suite Licensed Program Specifications* (GC31-6535)
- *z/OS V1R2.0 Communications Server: APPC Application Suite User's Guide* (SC31-8809)

Programming Publications

The following books contain information about programming:

- *Common Programming Interface Communications CPI-C Reference* (SC26-4399)
- *Communications Server for OS/2 Version 4 Application Programming Guide* (SC31-8152)

Other IBM Networking Publications

The following books contain information about other topics related to Communications Server for Linux:

- *SDLC Concepts* (GA27-3093)
- *Local Area Network Concepts and Products: LAN Architecture* (SG24-4753)
- *Local Area Network Concepts and Products: LAN Adapters, Hubs and ATM* (SG24-4754)
- *Local Area Network Concepts and Products: Routers and Gateways* (SG24-4755)
- *Local Area Network Concepts and Products: LAN Operating Systems and Management* (SG24-4756)
- *IBM Network Control Program Resource Definition Guide* (SC30-3349)

Index

Q

R

S

T

U

V

W

X

Communicating Your Comments to IBM

If you especially like or dislike anything about this document, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this document. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Please send your comments to us in either of the following ways:

- If you prefer to send comments by FAX, use this number: 1+919-254-4028
- If you prefer to send comments electronically, use this address:
  - comsvrcf@us.ibm.com.
- If you prefer to send comments by post, use this address:

```
International Business Machines Corporation
Attn: Communications Server for Linux Information Development
P.O. Box 12195, 3039 Cornwallis Road
Department AKCA, Building 501
Research Triangle Park, North Carolina 27709-2195
```

Make sure to include the following in your note:

- Title and publication number of this document
- Page number or topic to which your comment applies.

Program Number:

Printed in USA

SC31-6779-02