IBM Communications Server for Linux

Administration Guide

Version 6.2.3

SC31-6771-02

IBM Communications Server for Linux

Administration Guide

Version 6.2.3

SC31-6771-02

Note:
Before using this information and the product it supports, be sure to read the general information under Appendix C, "Notices," on page 179.

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 is a guide for enabling, configuring, and managing IBM® Communications Server for Linux®, an IBM software product that enables a computer running Linux to exchange information with other nodes on an SNA (Systems Network Architecture) 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 hardware on which Communications Server for Linux operates and with the Linux operating system. They must also be knowledgeable about the network to which the system is connected and understand SNA concepts in general.

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 Linux operating 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 guide explains how to enable, configure, and manage Communications Server for Linux.

Organization of This Book

This book is organized as follows:

- Chapter 1, "SNA Terms and Concepts," on page 1, provides an overview of SNA and APPN® (Advanced Peer-to-Peer Networking®) concepts.
- Chapter 2, "Administering Communications Server for Linux," on page 29, describes the Communications Server for Linux administration tools and explains how to prepare for Communications Server for Linux configuration, how to enable and disable the Communications Server for Linux software on a server, and how to use the Motif and the command-line administration programs.
- Chapter 3, "Basic Configuration Tasks," on page 53, explains how to perform basic configuration tasks for Communications Server for Linux servers, including configuring client/server operations, configuring the SNA node, and configuring message logging for Communications Server for Linux.
- Chapter 4, "Defining Connectivity Components," on page 59, explains how to configure connectivity for the Communications Server for Linux node.
- Chapter 5, "Configuring Dependent LUs," on page 75, explains how to configure dependent LUs (logical units) for LU types 0–3 and LU pools.
- Chapter 6, "Configuring APPC Communication," on page 79, explains how to configure APPC (advanced program-to-program communications).
- Chapter 7, "Configuring User Applications," on page 101, explains how to configure user applications.
- Chapter 8, "Configuring Passthrough Services," on page 103, explains how to configure passthrough services, which support communication between host systems and local systems that are not directly connected.
- Chapter 9, "Managing Communications Server for Linux from NetView," on page 115, explains how to use the Communications Server for Linux remote command facility (RCF) to manage Communications Server for Linux and run commands on Communications Server for Linux nodes from a host running NetView®.
- Chapter 10, "Managing Communications Server for Linux Client/Server Systems," on page 123, explains how to configure and manage IBM Remote API Clients.
- Appendix A, "Configuration Planning Worksheets," on page 149, provides configuration worksheets for Communications Server for Linux.
- Appendix B, "Configuring an Invokable TP from the Command Line," on page 171, provides information about the command-line utility that enables a user or the writer of a TP installation program to define an invokable TP.

Typographic Conventions

The typographic styles used in this document are shown in Table 1.

Table 1. Typographic Conventions

Special Element	Sample of Typography
Emphasized words	**back up files before deleting**

Table 1. Typographic Conventions (continued)

Special Element	Sample of Typography
Document title	*Communications Server for Linux Administration Guide*
File or path name	**/usr/spool/uucp/myfile.bkp**
Program or application	**snaadmin**
Parameter or Motif field	*opcode*; *LU name*
Literal value or selection that the user can enter (including default values)	`255`; `On node startup`
Motif button	**Status**
Motif menu	**Services**
Motif menu item	**Configure node parameters**
User input	**0p1**
Computer output	**CLOSE**
Command or Linux utility	**define_node**; **cd**
General reference to all commands of a particular type	**query_*** (indicates all of the administration commands that query details of a resource)
Option or flag	**-i**
Variable representing a supplied value	*filename*; *LU_name*; *user_ID*
Return value	`0`; `-1`
3270 key	ENTER
Keyboard keys	**Ctrl+D**; **Enter**
Hexadecimal value	`0x20`
Environment variable	`PATH`
Function, call, or entry point	`ioctl`
Programming verb	GET_LU_STATUS

Graphic Conventions

This symbol is used to indicate the start of a section of text that applies only to the AIX® or Linux operating system. It applies to Linux servers and to the IBM Remote API Client running on AIX, Linux, Linux for pSeries or Linux for System z.

This symbol is used to indicate the start of a section of text that applies to the IBM Remote API Client on Windows®.

This symbol indicates the end of a section of operating system specific text. The information following this symbol applies regardless of the operating system.

What Is New for This Release

Communications Server for Linux Version 6.2.3 replaces Communications Server for Linux Version 6.2.2, Communications Server for Linux Version 6.2.1 and Communications Server for Linux Version 6.2.

Releases of this product that are still supported are:

- Communications Server for Linux Version 6.2.2
- Communications Server for Linux Version 6.2.1

The following releases of this product are no longer supported:

- Communications Server for Linux Version 6.2
- Communications Server for Linux Version 6.0.1 (V6.0.1), which was available as PRPQ 5799–RQA or 5799–RXL.

Communications Server for Linux Version 6.2.3 operates with the IBM Remote API Client Version 6.3.1.0.

New Functions

The following functions have been added to Communications Server for Linux in this release:

- IPv6 addressing is now supported in addition to IPv4.
  - TN Server and Enterprise Extender (HPR/IP), which rely on IP connectivity, can communicate using either IPv4 or IPv6.
  - In a Client/Server deployment, Remote API Clients can communicate with servers using either IPv4 or IPv6. (IPv6 does not support UDP broadcasts, so each client must be configured with at least one server name.)
  - The NOF API, Motif administration program, and command-line administration program all accept either IPv6 colon-hexadecimal addresses or IPv4 dotted-decimal addresses.
- Communications Server for Linux normally includes the TCP/IP Information Control Vector (0x64) in a NOTIFY request to the host for a TN3270 session or for an LUA application on a Remote API Client. This control vector provides information about the client that can be displayed on the host console or used by the host (for example in billing), including the IP address of the client. If the client address is an IPv6 address but the host is running a back-level version of VTAM® that cannot interpret IPv6 addresses, the client address may be displayed incorrectly on the host console. In such cases you can disable this function by specifying `NO_TCPIP_VECTOR` in the *ptf_flags* parameter on the **define_node** command.
- The command-line administration program and the NOF API now provide a function to query the Remote API Clients that are currently using a particular server. A NOF application can also register to receive indications when clients connect and disconnect.
- Support is included for Connection Network Reachability Awareness, in conjunction with similar support on the host computer. This means that if a single route across a Shared-Access Transport Facility (SATF) is not available (for example because a single route through an IP router is disabled), alternative routes through this facility will be used where possible, before later retrying the failed route.

- Log filtering allows you to suppress multiple instances of the same log message, so that each message from a specified list is logged only once. This reduces the volume of information in log files so that you can concentrate on new or important log messages.

Functions That Have Been Retired

No functions have been retired in this release.

Related Publications

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

- *IBM System/390® Principles of Operation*, SA22-7201
- *IBM z/Architect Principles of Operation*, SA22-7832
- *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. SNA Terms and Concepts

This chapter defines Systems Network Architecture (SNA) terms and concepts that are important to understanding and using Communications Server for Linux. For information about Communications Server for Linux, its capabilities, and how it implements the different SNA concepts described, see *Communications Server for Linux Quick Beginnings*. If you are already familiar with SNA and Communications Server for Linux, you can begin with Chapter 2, "Administering Communications Server for Linux," on page 29.

This chapter is divided into the following parts:

- "Systems Network Architecture" provides a definition of SNA.
- "Basic SNA Concepts" on page 2 explains terms and concepts that apply to any SNA network.
- "Basic APPN Concepts" on page 12 explains terms and concepts that apply only to SNA networks that support Advanced Peer-to-Peer Networking (APPN).
- "Accessing Subarea Networks from APPN Networks" on page 27 introduces terms and concepts that apply to networks that combine SNA and APPN.

Note: This chapter is not intended as a complete reference to SNA concepts. Detailed information about SNA can be found in the SNA publications listed in "Related Publications" on page xv.

Systems Network Architecture

Systems Network Architecture (SNA) is an IBM data communication architecture that specifies common conventions for communicating among a wide variety of hardware and software data communication products. This architecture consists of two kinds of definitions: formats that define the layout of messages exchanged by network components, and protocols that define the actions that network components take in response to messages.

An SNA network is a collection of computers that are linked together and communicate using SNA.

Originally, SNA was designed to enable communications with a host computer. Each network or subnetwork was controlled by the host; other computers communicated directly with the host, but not with each other. This older, host-controlled style of network is often referred to as subarea SNA. SNA has since been extended to support direct peer-to-peer communications between computers in the network, without requiring a host. This newer, peer-level networking is APPN.

Many SNA networks have elements of both subarea and peer-to-peer networking. As networks migrate from subarea SNA to APPN, an APPN-capable host may act to control older systems while also acting as a peer to newer systems. Similarly, a single computer may access both peer computers (in an APPN network) and an older host; its communications with the host are controlled by the host, but its communications with other computers are peer-to-peer and do not involve the host.

Basic SNA Concepts

SNA defines the standards, protocols, and functions used by devices—from mainframes to terminals—to enable them to communicate with each other in SNA networks.

SNA functions are divided into a hierarchical structure of separate layers, each performing a specific set of functions. This division of network functions into layers enables network devices to share information and processing resources without having detailed information about each device on the network. A user at a workstation can communicate with another user without knowing anything about the physical devices on the network or the connections between those devices.

Network Types

SNA supports the following types of networks:

- A subarea network is a hierarchically organized network consisting of subarea nodes and peripheral nodes. Subarea nodes, such as hosts and communication controllers, handle general network routing. Peripheral nodes, such as terminals, attach to the network without awareness of general network routing.
- A peer network is a cooperatively organized network consisting of peer nodes that all participate in general network routing.
- A mixed network is a network that supports both host-controlled communications and peer communications.

Note: Linux systems running Communications Server for Linux can act as a peripheral node in a subarea network, as a peer node in a peer network, or both at the same time.

SNA Nodes

In SNA networks, a node is a Linux system or other device—with associated software components—that implements SNA protocols and has at least one communication path to another node in the network. Each node manages its end of the network communication paths, and uses SNA protocols to communicate with the node at the other end of each path.

Because subarea networks and peer networks define the relationships among nodes differently, they also use different terms for node types (to describe the roles that nodes play in the network).

Node Types in a Subarea Network

SNA subarea networks support the following node types:

- Subarea nodes control communication and network resources for all attached nodes. SNA classifies subarea nodes according to their capabilities and the amount of control they have over other nodes:
  - Type 5 nodes provide SNA functions that control network resources, support transaction programs, support network operators, and provide end-user services. Because these functions are often provided by host processors, type 5 nodes are also known as host nodes. The devices and resources controlled by a type 5 subarea node constitute the domain of that node.
  - Type 4 nodes provide SNA functions that route and control the flow of data in a part of the network. Because these functions are often provided by communication controllers, type 4 nodes are also known as communication controller nodes.

- Peripheral nodes serve subordinate roles in subarea networks. For example, a peripheral node can support 3270 emulation or dependent LU 6.2 communication. Peripheral nodes are devices such as distributed processors, cluster controllers, or workstations; they are also classified into type 2.0 and type 2.1 nodes:
  - Type 2.0 nodes are always controlled by a type 4 or 5 node. They cannot establish communication with other nodes without the participation of a type 4 or 5 node. Type 2.0 nodes are referred to as dependent nodes.
  - Type 2.1 nodes can act as dependent nodes, but they can also communicate directly with other type 2.1 nodes.

  Note: Linux computers running Communications Server for Linux can function as type 2.1 or type 2.0 nodes.

A type 4 or 5 subarea node to which a peripheral node is attached acts as a boundary node. It performs a boundary function by translating between the network addresses used by a subarea node and the local addresses used by a peripheral node.

A simple subarea network includes the following components:

Host A host is a mainframe computer compatible with the original IBM System/370™. A host is traditionally a type 5 node. However, Communications Server for Linux on System z runs on a host computer as a type 2.1 or 2.0 node.

Communication controller
A communication controller, also known as a front-end processor (FEP), is a separate processor attached to the host. It manages the host's communications with other computers.

Communications link
A communications link connects the host site with an end-user site. The users are usually on a separate site from the host, so the two sites need to be connected by a communications link.

Terminal controller
At the remote end of the communications link is a terminal controller, also known as a cluster controller. It is responsible for controlling the use of the link, and routes data to the terminals. The most well-known IBM terminal controllers are the 3174 and 3274.

Terminals
Users run host applications or submit work to the host from terminals. The best-known IBM terminal is the 3270. A terminal can be connected through a terminal controller or directly connected to a communication controller.

Printers
Printers such as the IBM 3287 can also be attached to the terminal controller. They can receive output from the host.

As shown in Figure 1 on page 4, a diagram of a subarea network looks like an inverted tree.

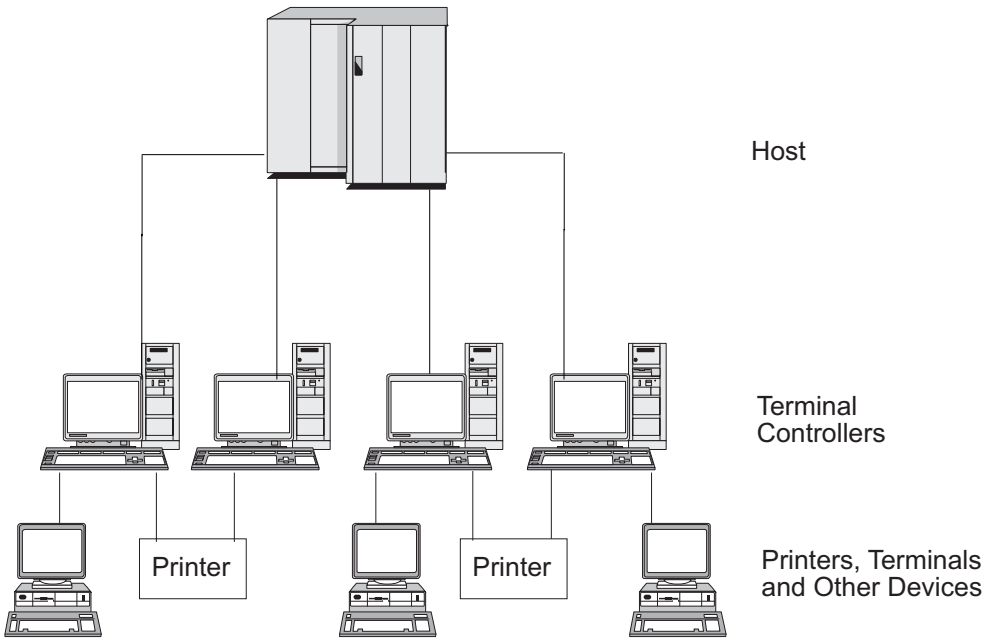

Figure 1. SNA Subarea Network

The root of the tree (at the top of the diagram) is the computer controlling the network. The branches are the communications links from the host to the other computers in the network (terminal controllers); the leaves (at the bottom of the diagram) are the terminals or printers attached to these computers, which are accessed by users.

The traditional subarea SNA set-up described here enables the users to use the resources of a single host system. The terminals provide only simple data entry and display functions to and from the terminal controller; the terminal controller is responsible for handling SNA communications between the terminals and the host.

The terminal controller and its terminals can be replaced by an SNA node using a product such as Communications Server for Linux. From the host's point of view, the node appears as a terminal controller. However, it provides the users with additional functions, such as the ability to access more than one host system and facilities for customizing screen displays. In addition, Communications Server for Linux runs on Linux computers that can also be used for other tasks not related to SNA (unlike the terminal controller, which is used solely for communications with the host).

Node Types in a Peer Network

Peer networks do not classify nodes hierarchically, as is done in a subarea network. Exchanges with other nodes are not controlled by a host or other centralized processor. Instead, any node can establish communication with any other node.

A peer network is composed of type 2.1 nodes. The nodes in a peer network can serve the following roles:

- APPN network nodes (NNs) identify the locations of network resources, determine routes for sessions between these resources, route sessions, and serve end nodes (EN) and low-entry networking (LEN) nodes directly attached to the network node. The domain of an APPN network node consists of itself and any end nodes for which it provides network services.

- APPN end nodes can access remote resources without requiring that those resources be configured on the end node. An end node can communicate with adjacent nodes on its own, but requires the services of a network node server to access nonadjacent nodes. The domain of an APPN end node includes only itself.
- APPN branch network nodes allow the APPN network to be separated into branches to simplify its topology and reduce network management overheads. They provide network node functions to end nodes in a branch separated from the main APPN network, while acting as end nodes in the main network itself. For more information, see "Branch Extender" on page 26.
- Low-entry networking nodes (LEN nodes) are type 2.1 nodes that do not support APPN functions. They can communicate with adjacent nodes in an APPN network, but do not participate in the APPN network. In a LEN node, all potential sessions with remote LUs must be predefined, either specifically or through a single default entry indicating that all remote LUs reside in an adjacent network node that can be accessed using a certain link. The domain of a LEN node includes only itself.

For more information about peer-oriented node types, see "APPN Node Types" on page 13.

Connectivity

For two nodes to communicate, each node must have a combination of hardware and software that supports data flow between the nodes. The hardware component consists of an adapter at each node and the transmission medium that connects the two adapters. The software component provides control of the hardware and the data exchanged over it.

Each node connected to a network has one or more link stations, which are the hardware and software in a node that control data flow to a specific adjacent node. To establish communication between two adjacent nodes, one of the link stations must first activate the link between the nodes.

Transaction Programs

Programs that exchange information across the SNA network are called transaction programs (TPs).

Following are examples of application programs that can include SNA TPs:

- Emulation programs
- File transfer
- Database transaction processing
- Network management
- Centralized data services

The TP accesses the network through a logical unit (LU) that establishes and maintains a session with a partner LU on another node. For more information about logical units, see "Logical Units" on page 6.

Note: Communications Server for Linux includes sample TPs for most supported APIs. For more information on sample TPs, refer to the programmer's guide for the API. You can also purchase SNA TPs as part of other products or create your own TPs (see "Application Programming Interfaces" on page 6).

Application Programming Interfaces

SNA TPs are written using application programming interfaces (APIs). APIs provide specific subroutines that enable SNA TPs to access SNA functions, such as those for exchanging data and performing control functions. These subroutines enable an SNA TP to communicate with another SNA TP on a remote node.

Communications Server for Linux includes the following APIs on all platforms:

- APPC—LU type 6.2 only
- CPI-C (Common Programming Interface for Communications)—LU type 6.2 only
- CSV (Common Service Verb) API
- LUA API

In addition, Communications Server for Linux includes the following proprietary programming interfaces:

- MS (Management Services) API (only for AIX or Linux systems)
- NOF (Node Operator Facility) API

Network Accessible Units

Communication between a TP and the SNA network occurs through network accessible units or NAUs (formerly called "network addressable units"), which are unique network resources that can be accessed (through unique local addresses) by other network resources.

SNA provides the following types of NAUs:

- Physical units (see "Physical Units")
- Logical units (see "Logical Units")
- Control points (see "Control Points" on page 8)

Note: Because TPs are considered users of the network, not components, they are not classified as NAUs.

Physical Units

Each SNA node contains a physical unit (PU). The PU manages resources (such as link resources) and supports communication with a host.

Note: On type 2.1 nodes (which can be APPN nodes), the control point provides PU services in addition to providing other services (see "Control Points" on page 8). Two type 2.1 nodes (such as Communications Server for Linux nodes) can communicate directly, without requiring the services of a host to establish communications.

Logical Units

Each SNA node contains one or more logical units (LUs). An LU provides a set of functions that are used by TPs and end users to provide access to the network. LUs communicate directly with local TPs and devices.

SNA defines several types of LUs, each optimized for a specific class of applications. LUs of different types cannot communicate with each other, but LUs of the same type can communicate even though they reside on different kinds of systems.

For example, a TP running on a Linux system can communicate with a TP on an AS/400® computer as easily as it can with a TP on another Linux system, as long as both TPs use the same LU type.

Communications Server for Linux supports the following LU types:

LU 6.2 (for APPC, 5250, APPC Application Suite, and CPI-C)
LU 6.2 supports program-to-program communication in a distributed data processing environment. The LU 6.2 data stream is either an SNA general data stream (GDS), which is a structured-field data stream, or a user-defined data stream. LU 6.2 can be used for communication between two type 5 nodes, a type 5 node and a type 2.0 or 2.1 node, or two type 2.1 nodes. (Type 2.1 nodes can serve as APPN nodes.)

This LU type provides more functions and greater flexibility than any other LU type. Unless you are constrained by existing hardware or software, LU 6.2 is the logical choice when developing new applications.

Note: Only LU 6.2 can provide independent LU functions.

LU 3 (for 3270 printing)
LU 3 supports application programs and printers using the SNA 3270 data stream.

For example, LU 3 can support an application program running under Customer Information Control System (CICS®) and sending data to an IBM 3262 printer attached to an IBM 3174 Establishment Controller.

LU 2 (for 3270 displays)
LU 2 supports application programs and display workstations communicating in an interactive environment using the SNA 3270 data stream. Type 2 LUs also use the SNA 3270 data stream for file transfer.

For example, the LU 2 protocol can support 3270 emulation programs, which enable workstations to perform the functions of IBM 3270-family terminals. In addition, LU 2 is used by other programs to communicate with host applications that normally provide output to 3270 display devices. Such TPs enable the workstation to achieve a form of cooperative processing with the host.

LU 1 (for SCS printing and RJE)
LU 1 supports application programs and single- or multiple-device data processing workstations communicating in an interactive, batch-data transfer, or distributed data processing environment. The data streams used by LU type 1 conform to the SNA character string or Document Content Architecture (DCA).

For example, LU type 1 can support an application program running under Information Management System/Virtual Storage (IMS/VS) and communicating with an IBM 8100 Information System. This enables an operator to correct a database that the application program maintains.

Applications that use LU 1 are often described as remote job entry (RJE) applications.

LU 0 (for LUA)
LU 0, an early LU definition, supports primitive program-to-program communication. Certain host database systems, such as IMS/VS (Information Management System/Virtual Storage) and some point-of-sale systems for the retail and banking industries (such as the IBM 4680 Store

System Operating System) use LU 0. Current releases of these products also support LU 6.2 communication, which is the preferred protocol for new applications.

Note: For information about the data streams used by SNA logical units, refer to *Systems Network Architecture Technical Reference*.

Control Points

A control point (CP) is an NAU that manages network resources within its domain, controlling resource activation, deactivation, and status monitoring. The CP manages both physical resources such as links, and logical information such as network addresses.

SNA defines the following types of network control points:

System services control point
On a type 5 node, the CP is called a system services control point (SSCP). It manages and controls the network resources in a subarea network. For example, an SSCP can use a directory of network resources to locate a specific LU under its control, and can establish communication between two LUs in its domain. An SSCP can also cooperate with other SSCPs to establish connectivity between LUs in different subarea domains.

The SSCP also provides an interface to network operators at the host system, who can inspect and control resources in the network.

Physical unit control point
On type 4 nodes and type 2.0 nodes in a subarea network, the control point is called a physical unit control point (PUCP).

Control point
On type 2.1 nodes, the control point provides both PU and LU functions, such as activating local link stations, interacting with a local operator, and managing local resources. It can also provide network services, such as partner LU location and route selection for local LUs.

In a subarea network, the CP on a Communications Server for Linux node acts as a type 2.0 PU. It communicates with an SSCP on a host and does not communicate with other CPs in the subarea network.

When participating in an APPN network, the CP exchanges network control information with the CPs in adjacent nodes. The CP can also function as an independent LU of type 6.2. The CP acts as the default LU for TPs on the local node. For more information about the APPN control point, see "APPN Control Point" on page 15.

Sessions

NAUs communicate with NAUs in other nodes over temporary logical communication channels called sessions. Before two TPs can communicate, their LUs must establish a session. The LU that manages the session on the local node is the local LU; the LU that manages the session on the remote node is the partner LU.

Session Types

Communications Server for Linux is primarily concerned with the following types of sessions:

LU-LU sessions
In order for two TPs to communicate, the LUs that support the TPs must

first establish an LU-LU session. In general, a session is established when a TP in one SNA node tries to communicate with a TP in another node and no existing session between the LUs on the two nodes is available.

SSCP-LU sessions
A dependent LU (see "Dependent and Independent LUs") must have an active SSCP-LU session with an SSCP on a type 5 node before it can have a session with an LU in the subarea network. Once an SSCP-LU session is active, a dependent LU can solicit an LU-LU session.

SSCP-PU sessions
Before an SSCP-LU session can be established, the PU controlling the LU must have an active SSCP-PU session with an SSCP on a type 5 node. The SSCP-PU session is used to pass control data and network management data between the PU and SSCP.

CP-CP sessions
In an APPN network, adjacent nodes establish CP-CP sessions. These sessions are used to search for a resource in the APPN network and to maintain topology information (see "APPN Control Point" on page 15).

Logical Unit Attributes for Sessions

Logical units have attributes that determine how they interact during LU-LU sessions. These attributes are determined by the architecture of SNA. LUs can be primary or secondary, and dependent or independent.

Primary and Secondary LUs: To establish a session, one LU requests session activation by sending a BIND request to another LU:

- A primary LU is the LU that sends the `BIND` request for a given LU-LU session.
- A secondary LU is the LU that receives the `BIND` request.

Peer networks do not use a fixed hierarchy of nodes and do not have predetermined primary or secondary LUs.

Note: In a peer network, an independent LU that is participating in multiple sessions (see "Multiple and Parallel Sessions" on page 10) can act as a primary LU for one session and a secondary LU in another.

Dependent and Independent LUs: All type 0, 1, 2, and 3 LUs are dependent LUs. Type 6.2 LUs can be configured as either dependent or independent LUs.

- A dependent LU (also known as an SSCP-dependent LU) requires the services of an SSCP to establish a session with another LU. An SSCP-LU session must be established before a dependent LU-LU session can be established.

  A dependent LU can be in session only with LUs on an SNA host. Because of this restriction, dependent LUs usually use subarea networks (also known as host-mediated networks). However, the dependent LU requester (DLUR) function enables session traffic from dependent LUs to flow over APPN networks. For more information about DLUR, see "Accessing Subarea Networks from APPN Networks" on page 27.

  A dependent LU on a peripheral node is always the secondary LU.
- An independent LU can establish sessions with other independent LUs without the aid of an SNA host. LU 6.2 is the only LU type that can be independent.

  An independent LU can act as a primary or as a secondary LU when establishing a session.

Multiple and Parallel Sessions

An independent LU can participate in sessions with more than one remote LU at the same time (multiple sessions).

An independent LU can also participate in parallel sessions, or multiple concurrent sessions with the same remote LU.

Dependent LUs (including dependent LU 6.2) cannot have multiple sessions.

LUs with multiple and parallel sessions are shown in Figure 2. LUA and LUB have parallel sessions. LUA also has multiple sessions: two with LUB and one with LUD. LUD has multiple sessions with LUA and LUC.

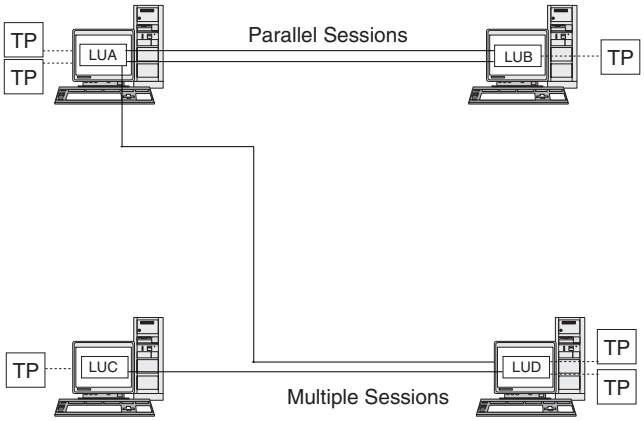

Figure 2. Multiple and Parallel Sessions

Conversations

This section applies to LU 6.2 only.

Once a session is established between two LUs, the LU-LU session supports the exchange of information between two TPs, which have the exclusive use of the session to execute a transaction. This exchange of information is called a conversation. Only one conversation can use a particular session at a time, but sessions are serially reusable (many conversations can use the same session, one after another).

To initiate a conversation, a source TP sends a request to its LU, asking it to allocate a conversation with a remote TP. The invoking TP (or source TP) initiates the conversation, like the calling party in a telephone conversation. The invokable TP or target TP (the remote TP) is the partner in the conversation, like the party who receives a telephone call.

As shown in Figure 3 on page 11, information is exchanged between TPs and LUs to enable one node to communicate with another. Although the TPs appear to be communicating directly, the LUs on each node are the intermediaries in every exchange.

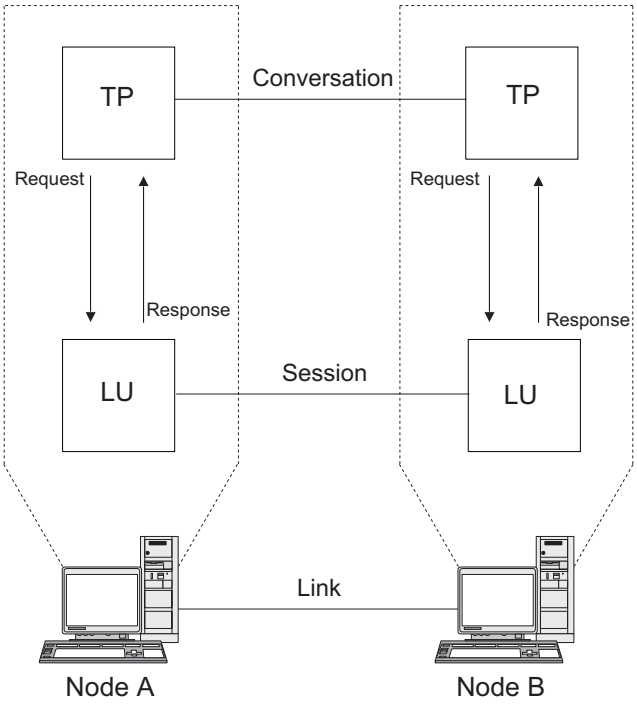

Figure 3. Communication between Transaction Programs and Logical Units

SNA defines two types of conversations: basic and mapped. These two types of conversations use different methods to indicate the length of transmitted or received data packages to be passed between Communications Server for Linux and the TP.

- In a basic conversation, data must be formatted by the TP as logical records before being presented to the SEND function.

  A logical record consists of a two- or four-byte header starting with a two-byte length field, often represented as "LL," followed by up to 32,765 bytes of data. Logical records can be grouped together and sent as a block, transmitting more than one logical record with a single call to the SEND function.
- In a mapped conversation, information is passed to the SEND function as a pointer to a single, unformatted block of data; the length of the block is passed as another parameter. The block cannot be received as one or more logical records; the receiving TP must do whatever record-level formatting is required.

Modes

Each LU-LU session has an associated mode that defines a set of session characteristics. These session characteristics include pacing parameters, session limits (such as the maximum number of sessions between two LUs), message sizes, and routing parameters.

Each mode is identified by a unique mode name. The mode name must be the same on all SNA nodes that use that mode.

Route Selection

To establish an LU-LU session, a route must be calculated between the nodes where the two LUs reside. A route is an ordered sequence of links and nodes that represents a path between the two nodes.

SNA networks support the following methods of route selection:

- For subarea networks, you must predefine all routes between subarea nodes.
- For peer networks that do not support APPN, type 2.1 nodes can support sessions only with adjacent nodes; their sessions cannot be routed through intermediate nodes.
- For APPN networks, SNA can compute routes dynamically at the time of session initiation, using a class of service specified for the mode used by the session (see "Class of Service").

The High-Performance Routing (HPR) feature of APPN provides the following functions:

- Rapid Transport Protocol (RTP) minimizes cycles and storage requirements for routing network layer packets through intermediate nodes on a session route.
- Automatic network routing (ANR) enables APPN networks to automatically reroute sessions if a portion of the originally computed route fails.

Class of Service

Class of service (COS) is a definition of the transport network (data link control and path control) characteristics—such as route security, transmission priority, and bandwidth—that the local node can use to establish a particular session. The COS definition assigns relative values to factors such as acceptable levels of security, cost per byte, cost per connect-time, propagation delay, and effective capacity.

In a subarea network, a COS is derived from the mode associated with a session, as defined in the host system.

APPN network nodes use the COS to compute session routes between independent LUs. For more information about session routing in APPN networks, see "Session Routing" on page 19.

Basic APPN Concepts

Advanced Peer-to-Peer Networking (APPN) is a network architecture that supports distributed network control. It makes networks easy to configure and use, provides centralized network management, and supports flexible connectivity.

An APPN network is composed of type 2.1 nodes. Each node in the network is connected by a link to at least one other node in the APPN network. CP-CP sessions are established over each of these links to adjacent nodes (nodes in the same network that can establish direct links without going through a third node). All of the nodes in an APPN network share a common network name.

APPN nodes can include processors of various sizes, such as the Application System/400® (AS/400), PCs running CS/NT, systems using Virtual Terminal Access Method (VTAM), and Linux servers running Communications Server for Linux.

APPN provides the following functions:

- Support for APPN network nodes and end nodes as well as non-APPN peer nodes (see "APPN Node Types" on page 13)
- APPN control point functions (see "APPN Control Point" on page 15)
- Directory services to support finding specific logical units (see "Locating Resources" on page 15)

- Topology and routing services to support session establishment using intermediate session routing (ISR), automatic network routing (ANR), or connection networks (CNs) (see "Session Routing" on page 19 and "APPN Connection Networks" on page 25)

Note: An APPN node can also be connected to a subarea network, serving as both an APPN node in a peer network and a peripheral node in a subarea network.

APPN Node Types

The following types of nodes can be part of an APPN network:

- Network nodes (see "APPN Network Nodes" on page 14)
- End nodes (see "APPN End Nodes" on page 14)

In addition, low-entry networking (LEN) nodes can be connected to an APPN network, but they do not use APPN features (see "LEN Nodes" on page 14).

A sample APPN network is shown in Figure 4.

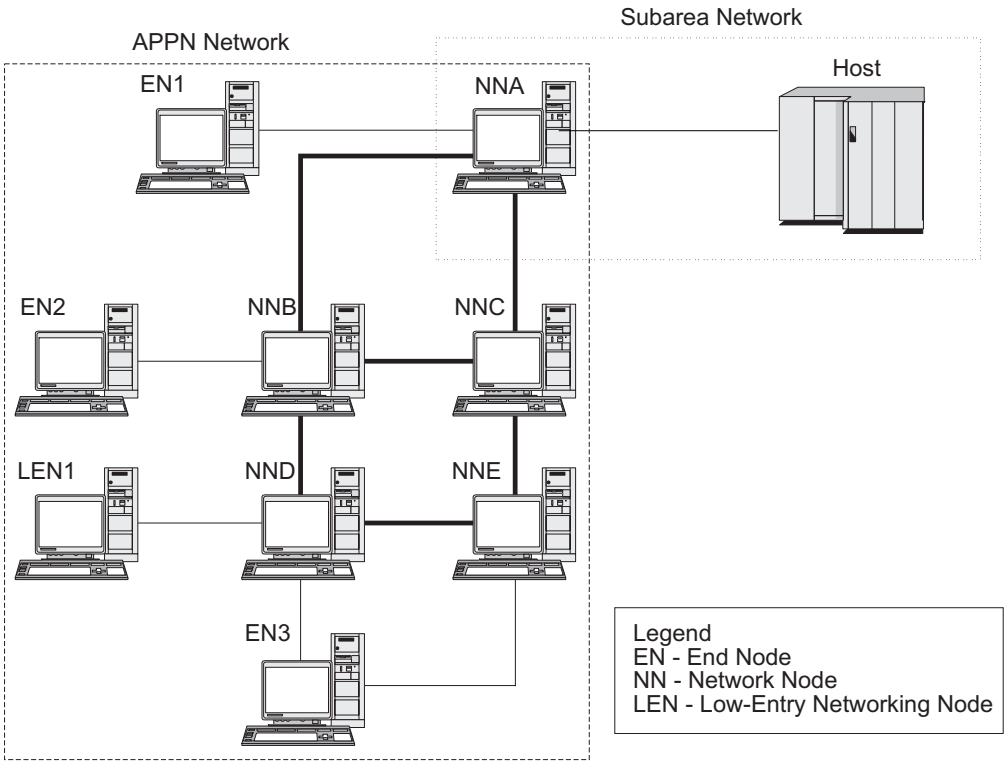

Figure 4. Portion of a Sample APPN Network

This example shows an APPN network that includes five network nodes (NNs), three end nodes (ENs), and a LEN node. The network nodes form the backbone of the APPN network; end nodes access the network through the network nodes. LU 6.2 TPs on any node can communicate with any other LU 6.2 TPs in the network.

One of the APPN network nodes (NNA) also participates in a subarea network, connecting to a host through a communication controller. This node functions as an APPN node when communicating with nodes in the APPN network, and as a peripheral node when communicating with nodes in the subarea network. Through

this network node, LU type 6.2 LUs on other nodes in the APPN network can establish LU-LU sessions with LU type 6.2 LUs on the host.

APPN Network Nodes

An APPN network node is a type 2.1 node that provides distributed directory and routing services for all LUs in its domain. These LUs can be located on the network node itself, or on an APPN end node or LEN node for which the network node provides services. Because an APPN network node acts as the network entry point for end and LEN nodes in its domain, the network node is also referred to as the network node server for those nodes.

A network node provides the following services:

- LU-LU session services for its local LUs
- Directory searches and route selection for all LUs in its domain
- Intermediate session routing (see "Intermediate Routing" on page 22)
- Routing for management services (MS) data, such as alerts, between a served end node and an MS focal point

APPN End Nodes

An APPN end node is a type 2.1 node that serves as an end point in an APPN network. It maintains directory information only for local resources. An APPN end node can independently establish sessions between local LUs and LUs on adjacent nodes. For sessions with LUs on nodes not directly connected to the end node, an end node requests routing and directory information from its network node server using CP-CP sessions.

APPN end nodes can register their local LUs with their network node server. This capability means the network operator at the network node server does not have to predefine the names of all LUs on the attached end nodes to which the network node provides services.

An APPN end node can be attached to multiple network nodes (see EN3 in Figure 4 on page 13), but it can have CP-CP sessions active with only one network node at a time—its network node server. The other network nodes can be used only to provide intermediate routing for the end node or as substitute network node servers if the main network node server becomes unavailable.

An APPN end node can also have a direct link to another APPN end node or a LEN node, but CP-CP sessions are never established between two end nodes.

LEN Nodes

A low-entry networking node (LEN node) is a type 2.1 node that uses independent LU 6.2 protocols, but does not support CP-CP sessions. It can be connected to an APPN network node or end node, but does not support APPN functions.

An APPN network node can provide routing services for an attached LEN node, enabling the LEN node to participate in an APPN network without requiring link stations to be defined between the LEN node and all of the nodes in the APPN network.

LUs in the APPN network with which the LEN node may want to establish sessions must be defined to the LEN node as if they reside on the LEN node's network node server. The LEN node establishes sessions with LUs on its network node server. The network node routes the session through the APPN network to the node in the network where the LU actually resides. LUs on the LEN node must

be predefined to the network node that serves the LEN node. LU resources on LEN nodes (unlike those on end nodes) cannot be registered on the network node server.

An APPN end node cannot provide intermediate routing. When a LEN node's only link is to an APPN end node, the LEN node can communicate only with LUs on the end node through the direct link between the two nodes.

APPN Control Point

An APPN control point is a set of functions that manages node resources and supports both physical unit and logical unit functions on a type 2.1 node. An APPN CP directs local node functions (such as activating and deactivating adapters and links), provides directory and topology information, and assists LUs in session initiation and termination.

Adjacent nodes in an APPN network use a pair of parallel CP-CP sessions to exchange network information and to provide directory and route selection services. Both sessions of a given pair must be active in order for the partner CPs to begin and sustain their interactions. Different node types use these sessions differently, as follows:

- Two parallel CP-CP sessions are established between an APPN network node and each adjacent network node. These CP-CP sessions are used to exchange directory, topology, and management services data.
- Two parallel CP-CP sessions are established between an APPN end node and the adjacent network node acting as the server for the end node. These CP-CP sessions are used to exchange directory, topology, and management services data.
- LEN nodes do not support CP-CP sessions.

The functions provided in CP-CP sessions vary based on the types of nodes involved, as follows:

- All CP-CP sessions conduct directory searches.
- CP-CP sessions between an end node and a network node provide the following functions:
  - Registering resources.
  - Routing management services data (such as alerts) between the end node and a focal point.
  - Routing topology data from each end node to its network node servers. This information can be used by the network node server to compute a route that does not flow through the network node server.
- CP-CP sessions between adjacent network nodes exchange topology information. As a result of this exchange, each network node creates an internal network topology database.

When setting up a node, you must define the CP name. The CP is also an LU that can support user sessions, and it can be the only LU defined in your node, if you so choose.

Locating Resources

To support communication between TPs, Communications Server for Linux first establishes a session between the logical units that control those TPs. APPN enables the CP on a node to locate LUs throughout the APPN network without requiring that the node have any configuration information for the remote LU. The

APPN function that dynamically locates LUs in the network is called directory services. Once a resource has been located, a route for the session is calculated through the APPN network.

Resource Names

Each node has a unique name consisting of two parts: a network name and a control point name. Together they constitute a fully qualified CP name. This name identifies each node to all other nodes in the network. Similarly, each logical unit is identified by a fully qualified LU name, consisting of a network name and LU name.

Note: For more information about network naming conventions, refer to *Communications Server for Linux Quick Beginnings*.

Directory Services

Each APPN node maintains a directory of network resources. Directory services is the component of the node CP that manages the local directory database and, in a network node, searches for network resources throughout an APPN network.

When the node is initialized, it includes the following information:

- Node type (APPN network node, APPN end node, or LEN node)
- Network ID of node
- CP name of node

Each node directory maintains entries for resources (LUs and CPs), including each resource's fully qualified name, type, and registration status. The specific resources stored in each local directory depend on the node type:

- A LEN node maintains a directory that includes its own LUs. It must also be configured with directory entries for all of its possible partner LUs. LUs in the APPN network with which the LEN node may want to establish sessions must be defined to the LEN node as if they reside on the LEN node's network node server. The LEN node establishes sessions with LUs on its network node server. The network node routes the session through the APPN network to the proper node in the network.

  A LEN node can also use wildcards in a directory entry to specify multiple partner LUs that can be accessed over a specific link.
- An APPN end node maintains a directory that includes its own LUs. It can also be configured to store directory entries for partner LUs in adjacent nodes. This enables local LUs to establish peer-to-peer sessions with those LUs without using APPN functions.

  If a resource is not locally defined to an end node or currently cannot be reached by the end node, the end node sends a request to its network node server asking it to search the APPN network for the resource.
- An APPN network node maintains a directory that includes its own LUs and the end node and LEN node LUs in its domain. An end node can dynamically register its LUs with its network node server. (LEN nodes cannot register LUs with a network node server, so LEN node LUs must be configured on their network node server.) A network node directory can also contain cached entries for LUs that are not in the network node's domain, but whose location has been determined through a previous search.

  Network nodes provide directory services to other nodes in two ways:

  – Searching for remote resources in response to session requests from end nodes or LEN nodes

– Responding positively to directory search requests from other network nodes when a named resource is found in the local directory

LEN Node Directories: An example of a LEN node directory is shown in Figure 5. Since LEN nodes do not support CP-CP sessions, the directory for Node LEN1 must contain all the LUs with which it communicates. The directory for Node LEN1 identifies its network node server (NNA) as the location for any LUs that are not on an adjacent peer end node. Since Node LEN1 can access the LUs only through Node NNA, it defines the CP on the network node as the "owning CP" of all the LUs, including LUs located on the end nodes.

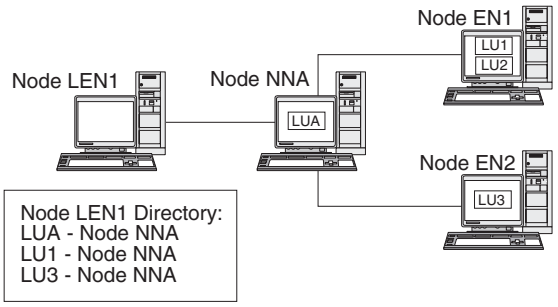

Figure 5. LEN Node Directory

To establish a session with an LU on a node that is not directly attached, Node LEN1 sends an LU-LU session activation (BIND) request to its network node server (Node NNA). The server automatically locates the destination LU and forwards the BIND.

Note: In this example, Node LEN1 can establish a session with LU1 on Node EN1 through its network node server, NNA. However, LU2 on Node EN1 is not defined in the directory for Node LEN1, so Node LEN1 cannot establish sessions with that LU.

End Node Directories: When an LU is not represented in an end node directory, the end node initiates a `LOCATE` search to find the desired LU. To activate the search for a remote LU, the end node invokes the services of its network node server. An example of an end node directory is shown in Figure 6 on page 18.

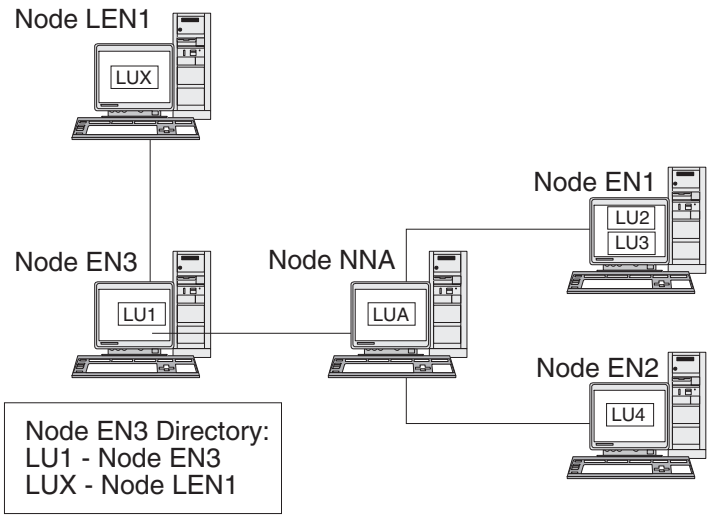

Figure 6. End Node Directory

Potential partner LUs in the APPN network do not need to be defined to the end node. However, in order for Node EN3 to establish a session with LUX on Node LEN1, the LU on the LEN node must be configured as a partner LU on Node EN3.

Network Node Directories: A network node provides distributed directory services to the end nodes it serves.

An example of a network node directory is shown in Figure 7.

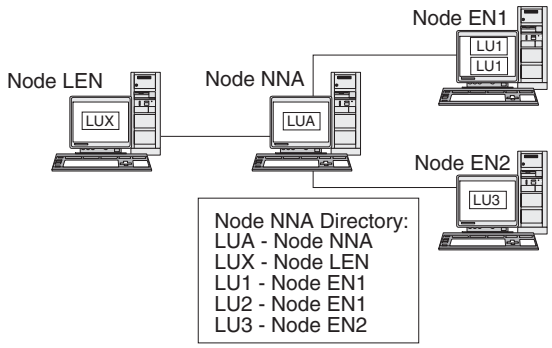

Figure 7. Network Node Directory

A network node locates a remote LU as follows:

1. The network node receives a request to locate an LU. The request can be any of the following:
   - The name of a destination LU sent by an end node or a LEN node to its network node server
   - An LU name specified in a `LOCATE` search request from an end node
   - An LU name specified in a `BIND` request from a LEN node
   - An LU name specified by a TP on the network node
2. If the destination LU is not located in the network node—but appears in its directory—the network node sends a directed search request to the destination network node server to verify the location of the LU.

If the LU is not in the network node directory, the node initiates a search of the network by sending a broadcast search to every adjacent network node.

3. Each node in turn propagates the broadcast and returns replies indicating success or failure.

For its future needs, a network node caches information obtained from successful broadcast searches.

An APPN end node can also receive (and respond to) `LOCATE` search requests from its network node server to search for, or confirm the continued presence of, specific LUs in the end node.

Each APPN end node registers its LUs with its network node server by sending the network node a registration message. In this way, the network node maintains current directory information for the end nodes in its domain. A LEN node cannot register LUs with its network node server. Therefore, all LUs on the LEN node must be predefined, through configuration, to the network node server.

Session Routing

APPN supports the following dynamic route selection procedures:

- For sessions with adjacent nodes, direct session routing.
- For sessions that traverse one or more intermediate nodes, one of the following:
  - Intermediate session routing (ISR), which provides a route that does not change during the course of the session.
  - High-Performance Routing (HPR), which includes the Rapid Transport Protocol (RTP) and automatic network routing (ANR) facilities. RTP minimizes cycles and storage requirements for routing network layer packets through intermediate nodes on a session route, and ANR enables you to reroute session traffic around route failures or congestion.

The APPN functions that provide dynamic route selection are known as topology and routing services (TRS).

Topology and Routing Services

Each APPN node includes a topology database that stores information about other APPN nodes and about transmission groups, which are sets of links between a specific pair of nodes. The contents of the database for a specific node depend on the node type:

- All network nodes share a copy of the network topology database. This shared database includes information about all other network nodes—including network IDs, CP names, and other node characteristics—and about the transmission groups between each pair of network nodes. This database provides a complete view of the network backbone topology—the nodes and transmission groups that can be used for routing sessions between any pair of nodes in the network.

  In addition, the topology database on each network node contains local information about transmission groups from that network node to adjacent end nodes or LEN nodes.

  The network node uses the topology database to calculate routes for sessions between LUs in its domain and remote LUs, or to provide information to other network nodes to enable them to calculate session routes.
- Each end node has a local topology database with information about transmission groups from that end node to adjacent nodes.

The end node provides this information to its network node server as part of the request to locate an LU and calculate a session route to that LU. The network node server uses the end node topology information when calculating the session route for the end node. The end node uses this information when establishing sessions with predefined LUs on adjacent nodes. The end node topology database supports communication only with adjacent nodes.

Note:

1. APPN network nodes and end nodes also maintain topology information about links to a connection network (see "APPN Connection Networks" on page 25).
2. LEN nodes maintain local topology information. They do not forward this information to a network node server.

As shown in Figure 8 on page 21, network topology information is replicated at all network nodes, and local topology information is stored at network nodes and end nodes.

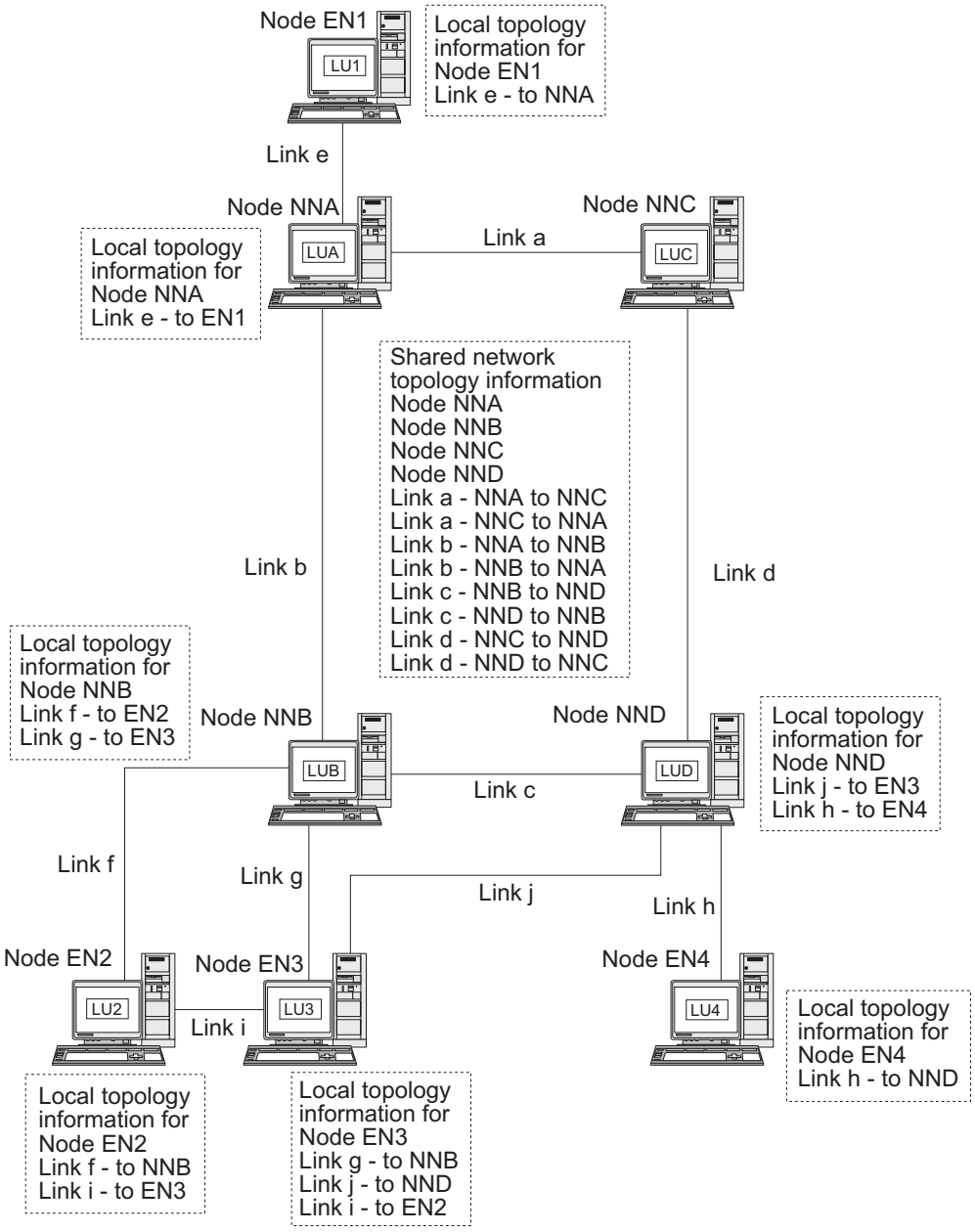

Figure 8. Network Topology Database in Network Nodes

The shared network topology database is duplicated at Nodes NNA, NNB, NNC, and NND. In addition, each of those nodes includes local topology information (except Node NNC, which does not have any local topology information because it does not have any links to end nodes). For example, Node NNB includes information for Link f to Node EN2 and Link g to Node EN3, but it does not include information for Link i, which connects Nodes EN2 and EN3.

End nodes include information only for links to adjacent nodes. For example, Node EN2 includes information about Link f to Node NNB and Link i to Node EN3.

Topology Database Updates: APPN network nodes use CP-CP sessions to exchange network topology information when a resource (such as a node or a link between two network nodes) is activated or deactivated, or when the

characteristics of an existing resource change. When such a change occurs, a network node generates a topology database update (TDU) that contains node identification, node and link characteristics, and update sequence numbers identifying the resource to be updated and the changes for the resource. Each TDU is sent to all active network nodes to ensure that the network topology database is kept current throughout the network.

Route Selection in an APPN Network: APPN directory services locates a specific session partner; topology and routing services calculates the optimal session route after the session partner has been located in the network. Each network node provides route selection services for sessions originated by its own LUs and by LUs at the end nodes or LEN nodes that it serves. A network node uses its own local topology information, plus information from the shared network topology database, to dynamically calculate routes between nodes.

Once the session partner has been located, the network node performs the following steps to select a route:

1. Obtains required characteristics for the session route.

   The LU requesting the session specifies a mode name that identifies session characteristics. The associated mode identifies a class of service that specifies requirements for the links used to route session traffic.
2. Obtains all transmission groups and network nodes for possible routes:
   - If the session request comes from an end node, the end node provides information about links it has to its network node server and to a connection network, if one exists.
   - If the session partner is not on an adjacent node, the network node server for the LU requesting the session uses the network topology database to identify network nodes and intermediate transmission groups in the route to the session partner.
   - If the session partner is on an end node, the end node (or its network node server) provides information about the link between the network node server and that end node (or the link between the end node and a connection network).
3. Excludes all network nodes and transmission groups that do not meet the specified characteristics for the session route.
4. Computes the optimal route for the session.

Depending on the specified class of service, the route calculation algorithm computes a weight value for each node and logical link and then totals the weights for each route. To select the optimal path, the network node computes the current least-weight route from the node containing the originating LU to the node containing the destination LU.

Intermediate Routing

Intermediate routing enables an APPN network node to receive and route data destined for another node. The origin and destination of the data can be an end node, another network node, or a LEN.

Intermediate routing supports sessions between LUs that are not on adjacent nodes. After a route has been selected for a session, APPN network nodes in the route use intermediate routing to forward session data to the next node in the route.

Resource characteristics maintained by the topology database can include congestion status. If a network node becomes heavily congested, the network node can relay this information to other network nodes in the network, making the congested network node less likely to be included in session routes calculated for new sessions.

APPN provides two types of intermediate routing:

- In intermediate session routing (ISR), available in all network nodes, the network node keeps track of each intermediate session. Each intermediate node adjusts the pacing of session data to control the rate at which data flows between adjacent nodes. Each intermediate node can also perform segmentation and reassembly of segmented data. In ISR, once a session route has been established, all data on that session uses the same route. If part of the route fails, the session ends.
- In automatic network routing (ANR), available in network nodes that support APPN's High-Performance Routing (HPR) function, intermediate network nodes can dynamically reroute session traffic if part of the route fails. ANR does not provide intermediate session pacing or segmentation and reassembly.

ANR enables intermediate nodes to route session traffic much faster than is possible with traditional APPN ISR. However, ANR requires additional overhead at the RTP (Rapid Transport Protocol) endpoints. In routes with few intermediate nodes, an ANR route might actually be slower than an ISR route, due to processing time at the endpoints. For routes containing a larger number of intermediate nodes (hops), ANR routes are typically faster. The exact location of the break-even point depends on the efficiency of the RTP nodes.

Direct Connectivity

Direct connectivity enables session traffic to travel directly between two nodes without the need for an APPN network node to route the session. In general, sessions between directly connected nodes can exchange data more quickly than sessions for which data is routed through a network node. For nodes on a shared-access transport facility (SATF)—for example, for nodes on a token ring as shown in Figure 9 on page 24—efficiency would be increased by defining links between every pair of nodes in your network. However, this can be a difficult task—the number of link stations is $n \times (n-1)$, where n is the number of nodes in the network.

An APPN network on a token ring is shown in Figure 9 on page 24.

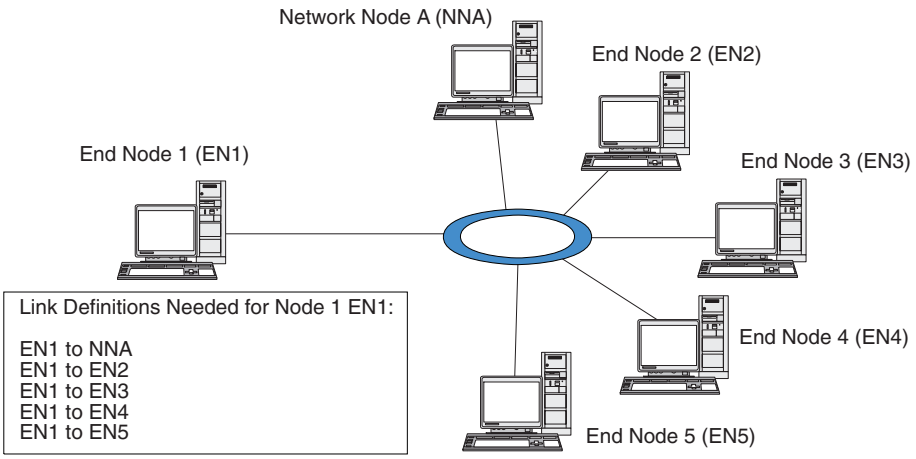

Figure 9. APPN Network Using a Shared-Access Transport Facility

If Node EN1 has a link definition for each of the links in the network, it can establish a direct link to any node. The link definitions needed to support direct links between Node EN1 and every other node in the APPN network are shown in Figure 10. For a network that includes five other nodes, Node EN1 needs five link definitions:

- EN1 to NNA
- EN1 to EN2
- EN1 to EN3
- EN1 to EN4
- EN1 to EN5

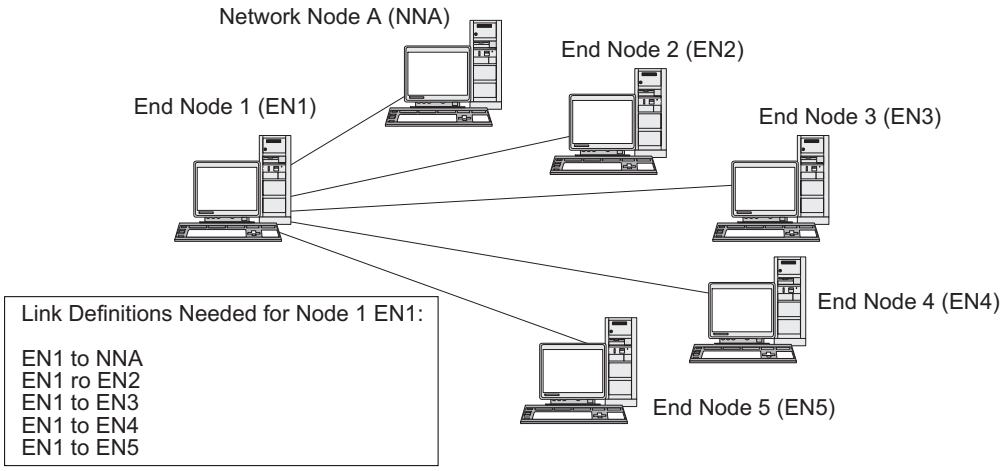

Figure 10. Definitions Needed for Direct Links from Node EN1 to Every Node in an APPN Network

If all of the nodes in the network are to support direct links to every other node, a total of 30 link definitions are needed on the six nodes in this example. In general, the number of link definitions can be calculated as $n \times (n–1)$, where n is the number of nodes in the network. In a larger network, the number of link definitions quickly becomes unwieldy. Increasing the number of link definitions between network nodes also increases the number of TDUs flowing through the network, which can degrade network performance.

APPN connection networks provide a solution to this problem.

APPN Connection Networks

For APPN networks attached to a shared-access transport facility (SATF), an APPN connection network greatly reduces the number of link definitions needed to support direct connectivity between nodes in the network. In a connection network, an APPN end node needs to configure only a single link to an adjacent network node server and a link to the connection network, instead of configuring every possible link to every node.

To use the connection network feature, an APPN network must meet the following conditions:

- The nodes in the APPN network must be linked using switched media such as token ring or Ethernet.
- All of the links in the APPN connection network must use the same media.
- The APPN network that contains the connection network must be fully connected. In a fully connected network, each node has at least one link that supports CP-CP sessions to an adjacent node.

In a connection network, the SATF serves as a virtual routing node (VRN) that attaches directly to each node in the connection network. The name of the connection network serves as the name of the control point for the VRN. The VRN supports the direct routing of session data between any two nodes in the connection network, but it does not establish CP-CP sessions with other nodes and it does not generate TDUs. Each node in the connection network requires only a link to its network node server.

The link definitions needed when using a connection network are shown in Figure 11. By using a virtual node, the connection network supports direct links between Node EN1 and every other node in the APPN network, yet it requires only two link definitions.

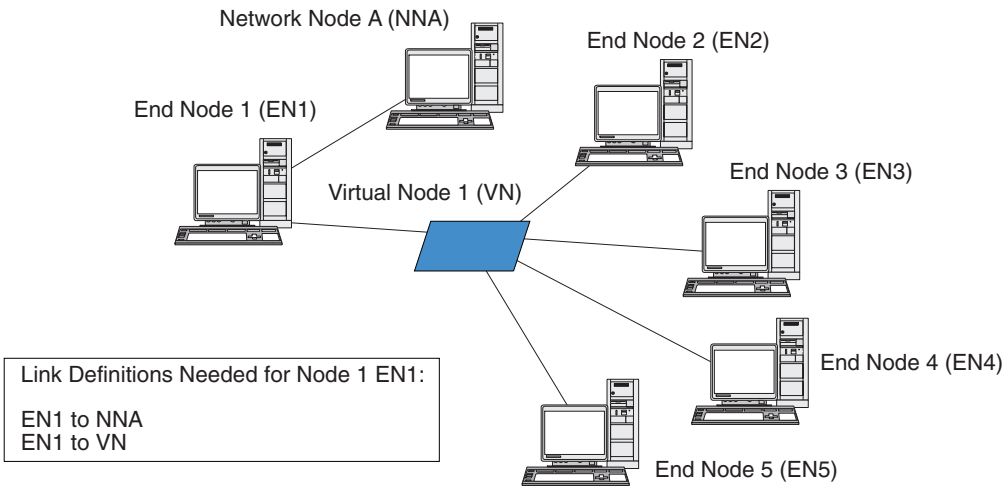

Figure 11. Definitions Needed for Direct Links Using a Virtual Node

To support direct links between any two end nodes in the APPN network, a total of ten link definitions is required. (Each end node needs two link definitions: one to a network node server and one to the virtual node.) Compared to the direct connectivity requirements for an APPN network that does not use a connection network (see Figure 10 on page 24), you can have a much smaller number of link

definitions (10 instead of 30 in this example). In a larger network, the difference in definition requirements becomes even more substantial.

A session between LUs on two nodes in the connection network is established as follows:

1. Each end node first establishes CP-CP sessions with its network node server. (If two end nodes have different network node servers, those network nodes must have a link that supports CP-CP sessions.)
2. End nodes also report their VRN links and local address information to the network node server. The local address information can be a service access point (SAP) address and a medium access control (MAC) address.
3. The server normally selects the direct link between two end nodes as the optimal route for the LU-LU session. It provides the node with the primary LU the information it needs to establish a dynamic link to the node with the partner LU.
4. The end nodes can then establish an LU-LU session without the need for intermediate session routing.

Branch Extender

As described in the previous sections, network nodes in an APPN network need to maintain topology information (about the location of other nodes in the network and the communications links between them), and to forward this information around the network when the topology changes. As the network grows in size, the amount of stored information and topology-related network traffic can become large and difficult to manage.

It is possible to avoid these problems by separating the network into subnetworks, so that each node only needs to maintain topology information about the nodes in its own subnetwork. However, this results in increased network traffic when trying to locate resources in other subnetworks.

The Branch Extender feature of APPN, illustrated in Figure 12 on page 27, provides a solution to these problems.

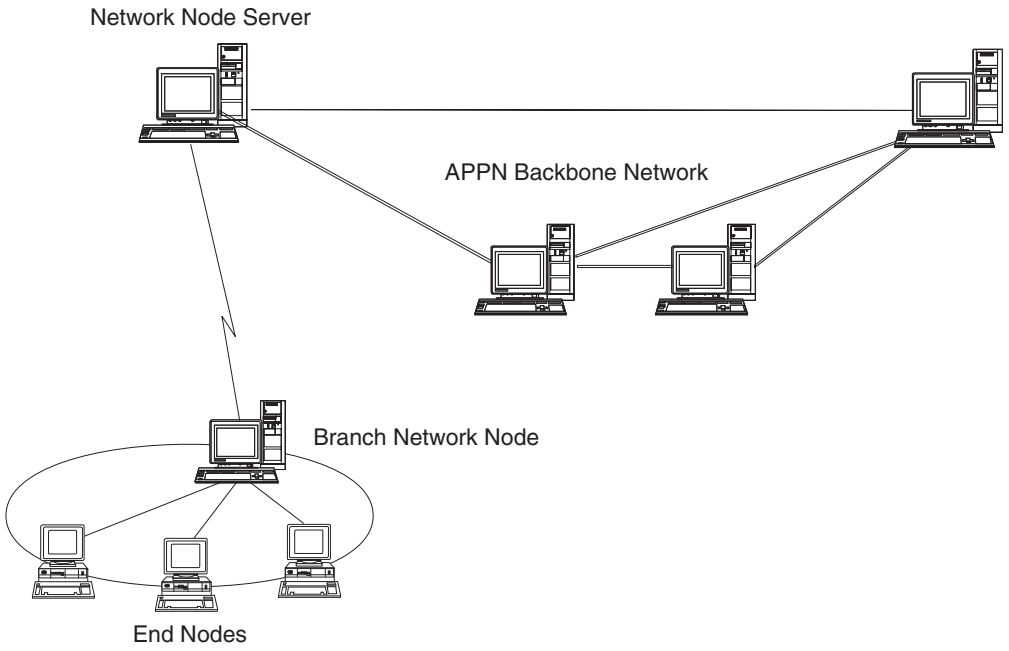

Figure 12. Branch Extender

As the name implies, Branch Extender is designed for networks that can be divided into distinct areas such as separate branches of a large organization. It works by separating out branches from the main backbone APPN network (for example, the network in the organization's headquarters).

Each branch contains a node of a new type called Branch Network Node (BrNN), which is connected to a Network Node in the main APPN backbone network. The BrNN combines the functions of an APPN network node and an APPN end node.

- To the backbone network, the BrNN appears as an End Node, connected to its Network Node Server (NNS) in the backbone network:
  - The nodes in the backbone network are not aware of the nodes within the branch, reducing the amount of topology information that must be stored.
  - Because the BrNN appears as an End Node, it does not receive topology information from the backbone network (topology information is transmitted only between Network Nodes).
  - The BrNN registers all resources in the branch with its NNS as though they were located on the BrNN itself. This means that the nodes in the backbone network can locate resources in the branch without having to be aware of the separate nodes in the branch.
- To the branch network, the BrNN appears as a Network Node, acting as the NNS for End Nodes in the branch. Each node in the branch sees the rest of the network as being connected through its NNS, in the same way as for a standard NNS.

Accessing Subarea Networks from APPN Networks

Although APPN networks do not require a host to control resources in the network, hosts often participate in APPN networks. APPN has been implemented on many host platforms, and allows the hosts to perform as network nodes in the APPN network while still providing an SSCP to control any old subarea SNA function.

Accessing Subarea Networks from APPN Networks

Many SNA networks contain elements of both subarea SNA and APPN. The backbone of the network is built from network nodes that must bridge the gap between a dependent LU and the facilities on the host. Two additional services are required to achieve this:

- Dependent LU server (DLUS) on the host provides access to the old SSCP functions and interfaces to the APPN network.
- Dependent LU requester (DLUR) on a network node or end node provides a means of transporting session traffic from dependent LUs to a host through an APPN network. This function enables dependent LU sessions to take advantage of the more versatile routing functions provided by APPN.

This combination of DLUR and DLUS (generally known simply as DLUR) allows dependent LU traffic to be transported over the APPN backbone. Existing SNA applications that use dependent LUs can be retained without modification, while taking advantage of APPN's network management, dynamic resource location, and route selection capabilities. In this way, DLUR provides a useful migration path from subarea SNA to APPN.

The dependent LU does not need to reside on the node that provides the DLUR function. If the DLUR function is provided by a network node, the dependent LU can be on an adjacent network node, end node, or LEN node. If the DLUR function is provided by an end node, the dependent LU must be on the end node itself.

Chapter 2. Administering Communications Server for Linux

For an overview of Communications Server for Linux administration and the different administration tools provided, see "Overview of Communications Server for Linux Administration."

The first step in administering Communications Server for Linux is configuring the node and its resources. Begin by planning for configuration as described in "Planning for Communications Server for Linux Configuration" on page 36.

Before you can configure Communications Server for Linux, you must enable the Communications Server for Linux software as described in "Enabling and Disabling Communications Server for Linux on the Local System" on page 37.

When Communications Server for Linux is enabled, you can run the Motif administration program (see "Using the Motif Administration Program" on page 40). The Motif administration program guides you through the configuration needed to support SNA communication using Communications Server for Linux. The Motif administration program is the recommended administration tool, because it minimizes the configuration information you need to provide and guides you through each step you must perform to support different types of communication (such as 3270 or APPC communication).

Alternatively, you can use the command-line administration program as described in "Using the Command-Line Administration Program" on page 50.

For each administration task, this guide provides information you can use for either Motif or command-line administration. Other configuration methods are discussed in "Administration Tools" on page 30.

Overview of Communications Server for Linux Administration

As the Communications Server for Linux administrator, you are responsible for installing the Communications Server for Linux software and for managing its resources. Before beginning Communications Server for Linux administration, you must understand the main features of the Communications Server for Linux product (see *Communications Server for Linux Quick Beginnings*). This section describes the administration tasks you must perform and the tools you can use to perform them.

Administration Responsibilities

To administer the Communications Server for Linux system, you need to do the following:

1. Define the resources of the Communications Server for Linux system, as required by the user programs that will be running. Work with the administrators of the host or peer computers with which Communications Server for Linux communicates, to ensure that the Communications Server for Linux configuration matches that of the remote system.
2. Initialize the Communications Server for Linux software.

3. Optionally, modify the configuration dynamically as your requirements change—by adding or removing resources, or by activating and deactivating the defined resources.
4. Monitor the status of active resources and gather diagnostics information to diagnose any problems that occur.
5. Optionally, create application programs or shell scripts to automate standard management operations.

These tasks are normally performed by a System Administrator at the site where the Communications Server for Linux system is installed. However, Communications Server for Linux also provides the service point command facility (SPCF), which enables an operator using the NetView program to perform Steps 2 and 3 remotely by issuing management commands at the NetView console. For more information about SPCF, see Chapter 9, "Managing Communications Server for Linux from NetView," on page 115.

Administration Tools

Communications Server for Linux provides a range of tools for administering the system. Depending on your requirements, you may not need to use all of them. This section summarizes the functions provided by each of these tools.

Note:

1. This document provides general information about Communications Server for Linux administration, which you can perform using any of the tools described in this section. For most purposes, the Motif administration program is recommended, because it provides context-sensitive guidance for node configuration and management.
2. For information about controlling who can use the Communications Server for Linux administration tools and the range of administration functions they can use, see "Administration Permissions" on page 35.

Communications Server for Linux includes the following administration tools:

- Motif administration program (see "Motif Administration Program").
- Command-line administration program (see "Command-Line Administration Program" on page 32, or refer to *Communications Server for Linux Administration Command Reference*).
- Service point command facility (see "Remote Command Facility" on page 32).
- Configuration files (see "Configuration Files" on page 33).
- Diagnostic tools (see "Diagnostic Tools" on page 34).

All of the Communications Server for Linux administration tools use the NOF API. You can also use that API to write your own administration tools. For more information, see "NOF Applications" on page 34.

Motif Administration Program

The easiest way to define and modify the Communications Server for Linux configuration is to use the Motif administration program (**xsnaadmin**). This program provides a graphical user interface from which you can view and manage Communications Server for Linux resources.

The following management operations are available:

- Defining Communications Server for Linux resources
- Starting and stopping a node and its connectivity resources

- Changing the configuration of defined resources
- Querying the configuration of defined resources and their current status if they are active
- Deleting resources

The Motif administration program can be used to manage both node resources (for any server on the LAN, as long as the Communications Server for Linux software is running on that server) and domain resources. For each type of communications (such as 3270 or APPC), the program guides you in setting up the configuration of the required resources.

Note: The windows and dialogs in the Motif administration program may differ from those shown in this guide, depending on the choices you make on a particular dialog.

The Motif administration program includes help screens that provide overview information for SNA and Communications Server for Linux, reference information for Communications Server for Linux dialogs, and guidance for performing specific tasks.

Before starting the Motif administration program, make sure the Communications Server for Linux software is enabled (for more information, see Chapter 2, "Administering Communications Server for Linux," on page 29). As with any X/Motif application, you may also need to set up the `DISPLAY` environment variable to indicate a suitable X server.

To start the Motif administration program in the background, issue the following command:

xsnaadmin &

All started Communications Server for Linux servers are shown on the main screen. For those that have already been configured, the program enables you to select a node, and then displays the selected node's configuration. Otherwise, the program prompts you to select a node and leads you through the required steps to define it.

For more information about how to use the Motif administration program to define and manage Communications Server for Linux resources, see "Invoking the Motif Administration Program" on page 40, or refer to the help screens provided by the program.

Note: The Motif administration program enables you to set up all required parameters for standard Communications Server for Linux configurations. For advanced parameters, the Motif administration program supplies default values. You need to supply only the essential configuration information, which enables you to set up SNA communications quickly and easily.

The other Communications Server for Linux administration tools, including command-line configuration, and NOF application programs, provide access to a wider range of configuration parameters and options than those shown in the Motif administration program. In most cases, however, you can perform all needed configuration from the Motif administration program, because it exposes the key fields you need to configure and hides the fields that most users should not need to modify. The default values supplied by

command-line configuration may differ from those supplied by the Motif administration program, because the Motif program can choose values more intelligently based on the context of the configuration task you are performing.

If you need to use these additional functions, you can still use the Motif administration program to set up the basic configuration, and use the other administration tools to specify the additional functions. When you later use the Motif administration program to manage the modified configuration, the program retains the changes you made using the other tools, although the additional functions you have configured are not displayed in the Motif program.

Command-Line Administration Program

The command-line administration program, **snaadmin**, enables you to issue commands to manage individual Communications Server for Linux resources. You can use **snaadmin** either directly from the Linux command prompt or from within a shell script.

Commands can be issued to a specific Communications Server for Linux node to manage the node's resources, to the SNA network data file to manage master and backup servers, or to the domain configuration file to manage domain resources.

All administration commands can be issued on a server. However, there are restrictions on which commands can be issued on an IBM Remote API Client.

- On Windows clients there is no **snaadmin** program, so no commands can be issued.
- On AIX and Linux clients you can issue any **query** or **status** command. Some other administration commands, defined in Communications Server for Linux Administration Command Reference, explicitly say that they can be issued from an IBM Remote API Client. Otherwise these commands are available only from a server.

You can get help for command-line administration by using any of the following commands:

- **snaadmin -h** provides basic help for command-line administration and usage information for command-line help.
- **snaadmin -h -d** provides a list of commands that can be supplied to the **snaadmin** program.
- **snaadmin -h** *command* provides help for the named *command*.
- **snaadmin -h -d** *command* provides detailed help for the named *command*, including a list of the configuration parameters that can be specified with the command.

Refer to *Communications Server for Linux Administration Command Reference* for more information.

Remote Command Facility

The remote command facility (RCF) provides the following facilities to support the administration of Communications Server for Linux from a NetView console on a host:

- Service point command facility (SPCF) enables an operator at a host NetView console to manage Communications Server for Linux from NetView by issuing Communications Server for Linux administration commands.

- UNIX® command facility (UCF) enables the NetView operator to issue standard Linux commands on the Communications Server for Linux computer.

For more information about RCF, see Chapter 9, "Managing Communications Server for Linux from NetView," on page 115.

Configuration Files

Configuration information for the Communications Server for Linux system is held in the following text files:

Node configuration file

The **/etc/opt/ibm/sna/sna_node.cfg** file contains information about Communications Server for Linux node resources for a specific node. This file resides on the computer where the node runs. It includes information about the node's resources and specifies which resources are active when Communications Server for Linux is started on the node.

This file provides an initial definition of the resources that are available; you can then use the other administration tools to modify the running node's resources as your requirements change. Any modifications you make are automatically saved to the file, so that the modified configuration can be used again when the node is stopped and restarted.

Domain configuration file

The **/etc/opt/ibm/sna/sna_domn.cfg** file contains information about Communications Server for Linux domain resources (resources not associated with a particular local node). The master copy of this file resides on the master server.

Invokable TP data file

The **/etc/opt/ibm/sna/sna_tps** file contains information that Communications Server for Linux needs to start invokable (target) TPs, and can also provide other information (such as the level of security required to access the TP). This file resides on the computer where the TPs run.

For more information about this file, see "Defining TPs" on page 86.

You can modify the configuration using the Motif administration program, the command-line administration program, or the NOF API. All of these tools make the required changes to the node configuration file or domain configuration file as appropriate. Because configuration information is stored as plain text, you can also modify the file directly using a standard ASCII text editor such as **vi**, or by means of a shell script using Linux utilities such as **awk** or **sed**. Any changes to configuration files using a text editor must be made **before** starting Communications Server for Linux. Refer to *Communications Server for Linux Administration Command Reference* for more information about Communications Server for Linux configuration file format.

Note: Communications Server for Linux configuration is a dynamic process; it is not necessary to define the entire configuration before starting the Communications Server for Linux software. The configuration file provides an initial definition of the available resources, but you can add, delete, or modify resources as necessary while the Communications Server for Linux software is running. Communications Server for Linux stores the current definition so that you can use it again when you need to restart the system.

The following files contain information about the Communications Server for Linux client/server network:

SNA network data file

The **/etc/opt/ibm/sna/sna.net** file contains information about which server is the master, and which servers can act as backup servers. This binary file resides on the master server. You can modify the contents of this file using the administration programs or the NOF API.

For more information about this file and how to modify it, see "Configuring Client/Server Functions" on page 53.

Client network data file

The **sna_clnt.net** file contains information about how to access Communications Server for Linux servers, required by an IBM Remote API Client. This text file resides on the client computer. You can modify the contents of this file using a standard ASCII text editor.

For more information about this file and how to modify it, see "Client Network Data File (sna_clnt.net)" on page 143. For information about configuring the equivalent information on a Windows client, see Chapter 10, "Managing Communications Server for Linux Client/Server Systems," on page 123.

NOF Applications

The Communications Server for Linux NOF API provides the same management functions as the command-line administration program, enabling you to define and manage Communications Server for Linux resources. This means that you can write your own application programs to administer Communications Server for Linux.

Refer to *Communications Server for Linux NOF Programmer's Guide* for more information.

Diagnostic Tools

Communications Server for Linux provides several diagnostics tools to help you diagnose and correct problems encountered during Communications Server for Linux operation:

- Any component detecting a problem or an exception (an abnormal condition that may indicate the cause of a problem) writes an entry to an error log file. In addition, all significant system events can be recorded in an audit log file. You can determine which types of events (problems, exceptions, or audits) are recorded. In a client/server network configuration, you can specify global settings for the types of events to record on all servers, and then override these on individual servers if necessary.
- Communications Server for Linux also maintains a usage log file, which is used to record information about the current and peak usage of Communications Server for Linux resources.
- You can specify the names and directories of the files used to hold each type of log information; if preferred, you can send both error and audit log information to the same file. On a client/server system, you can send messages from all servers to a central log file on one server (central logging), or send log messages to separate files on each server.
- Log files are generated as text files, and can be viewed using a standard ASCII text editor such as **vi**.
- You can choose full logging (which includes details of the cause of the log, and any action required, in the log file for each message), or succinct logging (which includes only a summary of the source of the log and the message text). When

using succinct logging, you can use the **snahelp** command-line utility to obtain the full cause and action text for a particular message number if you need further information.

- 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.
- For some error conditions, Communications Server for Linux sends a message to the Linux console to warn the operator, in addition to writing a problem message to the error log file.
- Many components can produce a trace file that records the activity of that component. Tracing degrades the performance of Communications Server for Linux components, and so is normally disabled.
- Using command-line utilities, you can filter trace files to extract specific information, and then format the trace information to interpret its contents or to produce a summary of message flows. The formatted output files can be viewed using a standard ASCII text editor such as **vi**.
- Communications Server for Linux can generate alerts and send them to the NetView program at a host computer. These alerts can be any of the following:
  - Link alerts from connectivity components, to provide information about connection problems
  - Alerts supplied by an application program using the MS API

Refer to *Communications Server for Linux Diagnostics Guide* for information about Communications Server for Linux log messages, using Communications Server for Linux trace facilities, and interpreting trace files.

For information about using the MS API, refer to *Communications Server for Linux MS Programmer's Guide*.

Administration Permissions

The Communications Server for Linux administration tools are intended for use by a restricted group of "SNA administrators" who have permission to manage SNA resources. To achieve this, the executable files are owned by the system administrator login `root` with a group ownership of `sna`. Only users who are members of the group `sna` can modify, start, or stop Communications Server for Linux resources; any user who is to have SNA administrator permissions must be a member of this group.

In the standard Communications Server for Linux installation, users who are not members of the group `sna` cannot run the Communications Server for Linux administration tools at all. If appropriate, you can allow these users to run the tools in read-only mode, so that they can view configuration and status information but cannot modify, start, or stop resources. To do this, use **chmod** to give read and execute permission for any user to the appropriate executable file or files:

Administration Tool	Executable File(s)
Motif administration program	**/opt/ibm/sna/bin/X11/xsnaadmin**
Command-line administration program	**/opt/ibm/sna/bin/snaadmin**

Any user can then run the appropriate administration tool and view information, but Communications Server for Linux will still prevent users not in the `sna` group from modifying, starting, or stopping resources.

Note: If you modify file permissions as described above, you will need to repeat this procedure after installing Communications Server for Linux PTFs or new releases.

Planning for Communications Server for Linux Configuration

Before you make any configuration changes it is very important to plan thoroughly. Changes that you make can cause disruption, not only to the users of your local node but possibly to users all around the network.

You may find it useful to draw a diagram of any changes that you are making to the topology of the network. If you are adding or removing connections to other nodes, draw a picture showing your node and the other nodes. You can use the Motif administration program to gather configuration information about all of the existing connections and add that information to your diagram.

When you add new resources to your diagram, it is easy to see whether they duplicate existing ones, or whether any names clash. Similarly, your diagram can help you decide which resources you need to remove and help you avoid deleting essential ones.

If you are configuring a Client/Server Communications Server for Linux system with more than one node, ensure that you include all the Communications Server for Linux nodes and their connectivity resources in your diagram.

Once you determine the changes you need to make, you can collect the configuration information that you need. You can use the task sheets in the online help files for the Motif administration program, or the planning worksheets described in "Planning Worksheets," to guide you in collecting configuration information for specific Communications Server for Linux functions.

Planning Worksheets

Before you begin to configure resources for Communications Server for Linux, gather all of the configuration data for the new resources. To record all of the information for a particular function or application that you need to support, use the planning worksheets in Appendix A, "Configuration Planning Worksheets," on page 149.

You will probably need to gather configuration information from several sources, such as network administrators, host administrators, application programmers, and end users.

If you are trying to connect to another node, the administrator at that node is a key contact. The administrator for a node can tell you names, addresses and characteristics of all the resources on that node. Often, you will need to ensure that matching configuration parameters are entered at the local node and the remote node.

Task Sheets

The online help screens in the Motif administration program contain task sheets that provide guidance for specific configuration tasks. The task sheets contain

pointers to all of the help screens for the dialogs that you will use to enter the configuration information. You can use these to browse the help and see exactly what data you must collect.

The task sheets also refer to more detailed help for each of the individual windows and dialogs that you must use to enter configuration information. Those help screens explain each field that you must fill in or select.

Enabling and Disabling Communications Server for Linux on the Local System

This section explains how to enable and disable the Communications Server for Linux software on the Linux server.

You must enable the Communications Server for Linux software before you can use any Communications Server for Linux tools (including the Motif administration program). Normally, the software is enabled automatically after you install Communications Server for Linux, but if necessary you can enable it manually.

Specifying the Path to Communications Server for Linux Programs

Communications Server for Linux executable programs are stored in a directory specific to Communications Server for Linux; when you run the programs, you need to specify the path to this directory. You can specify the path either by adding the directory to your PATH environment variable before you run the programs for the first time, or by including the directory name each time you run the programs.

The Motif administration program is stored in the directory **/opt/ibm/sna/bin/X11**, and the other programs are stored in the directory **/opt/ibm/sna/bin**. If you add these directories to the definition of the PATH environment variable in your **.login** or **.profile** file, Communications Server for Linux locates the programs automatically. Alternatively, you can specify the directory name when you run the program, as in the following examples:

/opt/ibm/sna/bin/sna start

/opt/ibm/sna/bin/snaadmin query_node

/opt/ibm/sna/bin/X11/xsnaadmin

The sample command lines shown in this manual assume that you have added the directories to your PATH environment variable, and do not include the directory names.

Enabling Communications Server for Linux Servers

This section describes how to enable Communications Server for Linux on a computer that was installed as a server (that is, with the SNA node components installed). If you are enabling Communications Server for Linux on a client, see "Enabling and disabling Remote API Clients on AIX or Linux" on page 143.

You must enable Communications Server for Linux on the local system before you can configure or manage the local node (either locally or from a remote Communications Server for Linux node).

Enabling and Disabling Communications Server for Linux on the Local System

To enable the Communications Server for Linux software, enter the following command at the Linux command prompt:

sna start [**-s**] [
-m *kernel_memory_limit*] [
-t]

Note: When you use the **sna start** command, the Communications Server for Linux software uses the directory from which you issued the command as its current working directory, and maintains one or more open file descriptors in that directory. This means that you will not be able to unmount the file system containing that directory while the Communications Server for Linux software is running. To avoid problems, you should start the Communications Server for Linux software from a directory on a filesystem that does not need to be unmounted; for example, you could use `cd /` to change to the root directory before using the **sna start** command.

When you install Communications Server for Linux, the installation utility automatically updates the startup file **/etc/rc.d/init.d/snastart** to include the **sna start** command. This ensures that Communications Server for Linux is started automatically at system startup. If you do not want Communications Server for Linux to be started automatically, you can remove or comment out this line, and then follow the instructions in this section to enable the Communications Server for Linux software manually.

The parameters and options for the **sna start** command are as follows:

-s Specifies that Communications Server for Linux should not write messages to the system console. If you do not use this option, Communications Server for Linux writes messages to the console when it ends, and also writes the text of certain error log messages to the console as well as to the log file.

-m *kernel_memory_limit*
Specifies the maximum amount of kernel memory, in kilobytes, that Communications Server for Linux should use at any time. (Kernel memory is used for internal data structures.) If a component of Communications Server for Linux attempts to allocate kernel memory that would cause the total amount of memory currently allocated to Communications Server for Linux components to exceed this limit, the allocation attempt fails.

If you do not use this option, kernel memory usage is not limited.

-t Activates tracing on all interfaces between kernel components, and also client/server tracing. (This option does not turn on DLC tracing.) Tracing enables you to diagnose problems that occur during startup. If you do not use this option, tracing is inactive at all interfaces; you can then activate it on specific interfaces as required, using the command-line administration program **snaadmin**.

Tracing on all interfaces degrades the performance of Communications Server for Linux components. After the software is enabled, you can use the command-line administration program **snaadmin** to stop tracing on any interfaces where it is not required. For more information about tracing, refer to *Communications Server for Linux Diagnostics Guide*.

Communications Server for Linux writes messages to standard error (normally your terminal's screen) to indicate that it is initializing, and to indicate whether initialization completes successfully.

If initialization fails, the messages include information about the cause of the error, and (where appropriate) additional information such as the Linux operating system error message. The text written to standard error may also include a message indicating that you can find further information in the error log file. The **sna start** command then ends with a nonzero exit code that indicates the nature of the error.

For more information about exit code values, refer to *Communications Server for Linux Diagnostics Guide*.

Advanced Options for the sna start Command

In some cases, particularly when you are testing out new Communications Server for Linux configurations, you may want to start Communications Server for Linux with a configuration that you have saved to a temporary file (rather than with the standard configuration in the files **/etc/opt/ibm/sna/sna_node.cfg** and **/etc/opt/ibm/sna/sna_domn.cfg**). To do this, you can use the following additional options on the **sna start** command:

-n *node_config_file*
-d *domain_config_file*

node_config_file is the full pathname of the file to which you have saved node configuration (instead of **/etc/opt/ibm/sna/sna_node.cfg**), and *domain_config_file* is the full pathname of the file to which you have saved domain configuration (instead of **/etc/opt/ibm/sna/sna_domn.cfg**).

Note: These options are not intended for general use. Do not use them unless you have a specific requirement to do so.

The **snagetpd** command will not operate correctly when Communications Server for Linux is running with these options, because it always collects information from the standard configuration files. Before using **snagetpd**, ensure that you are running with the standard configuration files by starting Communications Server for Linux without these options.

Disabling Communications Server for Linux Servers

Disabling the Communications Server for Linux software on a server automatically stops the Communications Server for Linux node and its associated connectivity components. Disabling Communications Server for Linux also stops any other processes (such as a 3270 emulation program) from using Communications Server for Linux resources on this server.

In general, you should stop individual services as users finish using them, and only disable the system when there is no Communications Server for Linux activity. Disabling the Communications Server for Linux software on a client stops any programs running on the client from accessing Communications Server for Linux facilities.

If you need to disable Communications Server for Linux while users are active, warn users that Communications Server for Linux is stopping, and give them time

to finish their activities before you disable the software. Use the Motif administration program or the command-line administration program to view details of active users.

If a 3270 emulation program is using LUs on the node when you disable the Communications Server for Linux software, all 3270 emulation sessions using these LUs end. The program continues to run, but the user cannot use the sessions until the software is re-enabled. Applications using the APPC, CSV, LUA, NOF, or MS APIs are notified by a `COMM_SUBSYSTEM_ABENDED` return code, and CPI-C applications by a `CM_PRODUCT_SPECIFIC_ERROR` return code.

To disable the Communications Server for Linux software, enter the following command at the Linux command prompt:

sna stop

If Communications Server for Linux is disabled successfully, **sna stop** returns an exit code of `0`. Any other exit code indicates that an error occurred and that the Communications Server for Linux software was not disabled. Refer to *Communications Server for Linux Diagnostics Guide* for more information about exit code values.

Using the Motif Administration Program

The Motif administration program provides a user-friendly interface for configuring Communications Server for Linux. This program is the recommended tool for administering Communications Server for Linux, because it guides you through the configuration process and minimizes the information you need to provide to create a workable configuration.

You can also use the Motif administration program to manage the Communications Server for Linux system while it is active. The administration program enables you to make and apply changes to the configuration while Communications Server for Linux is active. You can add, modify, and remove resources (in most cases, even when the node and its resources are active), and use the modified configuration immediately for continued operation.

The Motif administration program displays up-to-date status information through the same interface that is used for configuration, providing easy access to status information for both domain and node resources.

Alternatively, you can use Communications Server for Linux commands for configuration and system management. A summary of configuration and management commands is provided in "Using the Command-Line Administration Program" on page 50.

Invoking the Motif Administration Program

To use the Motif administration program for Communications Server for Linux, first make sure that Communications Server for Linux is enabled as described in "Enabling Communications Server for Linux Servers" on page 37. (As with any X/Motif application, you may also need to set up the `DISPLAY` environment variable to indicate a suitable X server.)

To start the Motif administration program running in the background, issue the following command:

xsnaadmin &

In a client/server environment, Communications Server for Linux displays the Domain window.

For a standalone system, Communications Server for Linux normally displays the Node window. However, if you have not yet configured the local node, it displays a help screen offering help with configuring the node for the first time.

Note: This guide uses the term window to describe Motif windows that display information about Communications Server for Linux resources. A window can contain one or more sections, or panes. A dialog is a Motif window on which you can enter information.

Resource Windows

The Domain window and the Node window show most of the information you need and provide easy access to additional information. From those windows, you can easily display information about resources in your local network.

The Domain window shows all defined nodes, and enables you to add, delete, start, and stop nodes. Double-clicking on any node brings up the Node window for that node.

The Node window shows all the key resources for a particular node.

The menus in the Domain and Node windows provide the following functions:

Selection
The functions in this menu relate to the node that is currently selected in the Domain window or the item that is currently selected in the Node window. From this menu, you can start or stop the node or zoom on it to display its Node window. When you select an item in the Node window, you can control, modify, or delete the item using controls in this menu, or add a new item in the currently selected pane.

Services
This menu provides easy access to all the dialogs required to configure the node for common tasks. Using this menu, you can add or modify resources or get help for configuration and management tasks.

Diagnostics
You can control logging and tracing from items in this menu.

Windows
You can easily access other windows from this menu. These windows include the following:

- LU Pools window
- CPI-C Destination Names window

Depending on the resources you select and the options you choose, the administration program can present additional resource windows, configuration dialogs, or status logs. You will also see context dialogs that enable you to select a specific resource to configure, confirmation dialogs that ask you to confirm a choice, and message pop-ups that provide feedback or error information. Each window and dialog also includes a help option.

Domain Window

The Domain window shows each active SNA node in the Communications Server for Linux domain for the system you are using. (A node does not appear in the Domain window if Communications Server for Linux is not running on the node.) Each node is identified using the name of the system. The Domain window also shows the current status of each node in the domain.

Note: If a server is unexpectedly missing from the list of nodes in the Domain window, verify that the server is switched on and that the Communications Server for Linux software is running on the server. If necessary, start the Communications Server for Linux software on that node using the **sna start** command (see "Enabling Communications Server for Linux Servers" on page 37).

One node in a domain is always identified as the configuration server for the domain. The Domain window shows the word "Master" next to that node. The Master configuration server always contains configuration information for domain resources. Backup configuration servers are identified by the word "Backup" on this window. Backup configuration servers contain copies of the configuration information for domain resources.

An example of a Domain window is shown in Figure 13.

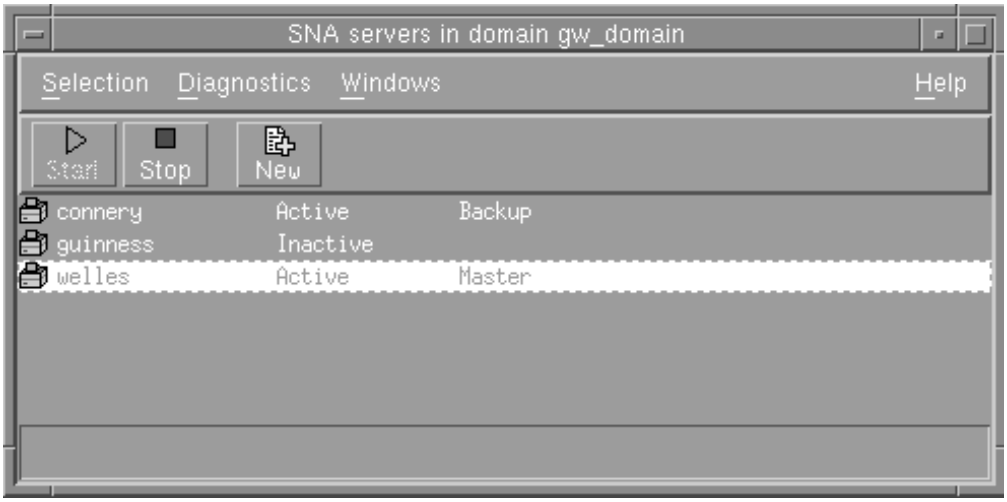

Figure 13. Communications Server for Linux Domain Window

If any active nodes in the domain (nodes on which Communications Server for Linux is running) are not configured, Communications Server for Linux prompts you to configure the node.

Note: The Domain window does not list IBM Remote API Clients. Clients use the resources of Communications Server for Linux servers (SNA nodes) to access SNA resources.

You can perform any of the following administration tasks from the Domain window:

Start or stop any node in the domain

Select the line for the node and click on the **Start** or **Stop** button on this window. (Alternatively, you can click on the line for the node, then select **Start node** or **Stop node** from the **Selection** menu.)

Administer a specific node

Double-click on the line for that node on the Domain window. (Alternatively, you can click on the line for the node, then select **Properties** from the **Selection** menu. You can also select the window for the node from the **Windows** menu.)

When you select a node to be administered, Communications Server for Linux displays the Node window as shown in Figure 14 on page 44. (For a standalone system, Communications Server for Linux does not display the Domain window, because the domain has only one node. Instead, Communications Server for Linux immediately displays the Node window when you start the administration program.)

Add a node to the list of servers for the domain

Click on the line for the node and select **Make configuration server** from the **Selection** menu.

Remove the node from the list of servers for the domain

Click on the line for the node and select **Remove configuration server** from the **Selection** menu.

Configure logging for all nodes in the domain

Select **Logging** from the **Diagnostics** menu.

Turn tracing for a specific node on or off

Click on the line for the node and select **Tracing on selected node** from the **Diagnostics** menu.

Get information about domain resources

Choose any of the options on the **Windows** menu. In addition to shared domain resources, the **Windows** menu also lists each Node window in the domain.

Node Window

A sample Node window is shown in Figure 14 on page 44. The title bar shows the name of the system.

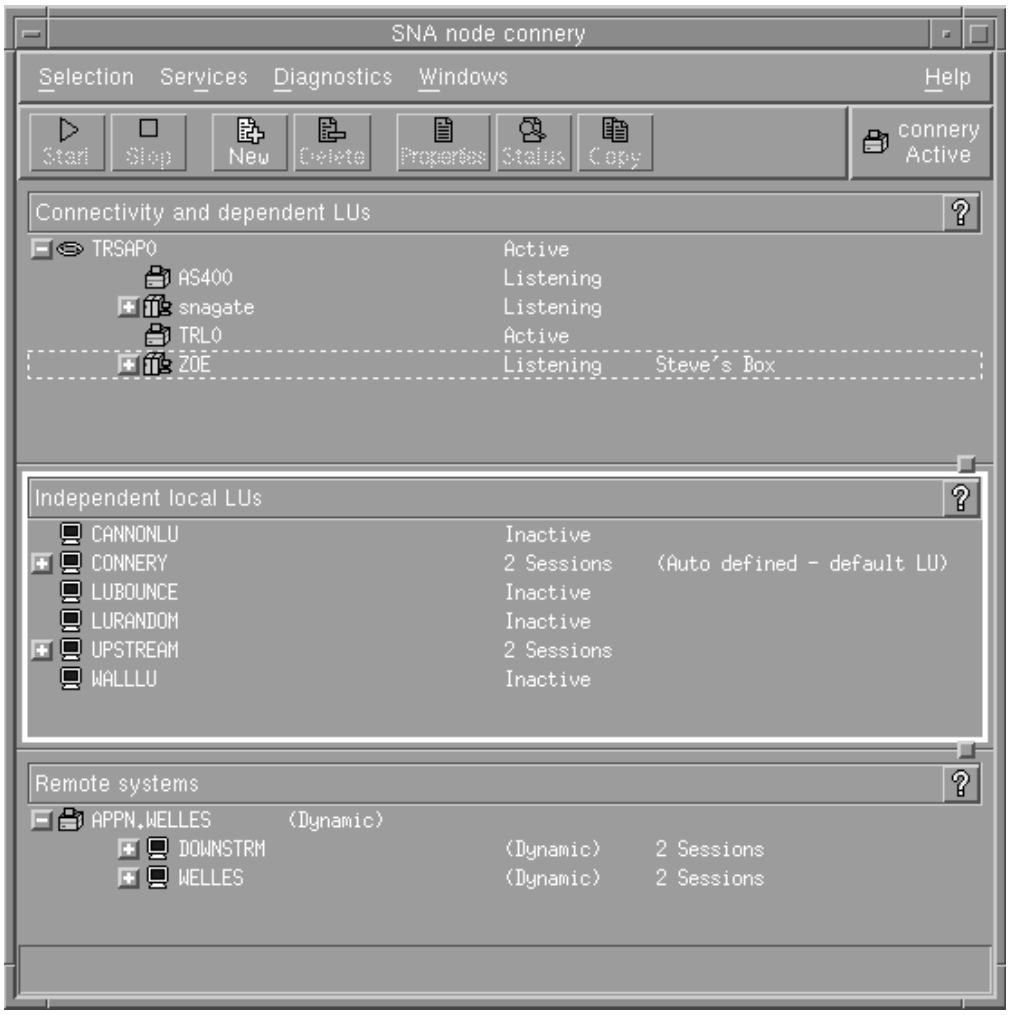

Figure 14. Node Window

From the Node window, you can add, delete, modify, and manage all of the resources and components for the Communications Server for Linux node. The layout of the resources in the window shows the relationships among resources and enables you to control which resources are displayed.

The Node box in the top-right corner of the Node window indicates whether the node is `Active` or `Inactive`.

Any ports, local LUs, and remote nodes that are defined on the node are always displayed. The Node window shows each link station below its parent port, and each dependent LU below its parent link station. It also shows partner LUs below local LUs and below remote nodes.

The body of the Node window is split into the following panes for the different types of resources for the node:

Connectivity pane

The top pane of the Node window lists connectivity resources for the node, including ports, link stations or PUs on each port, and dependent LUs on a specific link station or PU. For each resource, this window shows current status information.

Independent Local LUs pane
The middle pane shows independent LUs for the node. For each LU, this window also displays information about sessions using the LU.

Remote Systems pane
The lower pane shows information about remote nodes and partner LUs. It also shows session information for each remote node or partner LU.

To change the relative sizes of the panes, click and drag on the boundaries between panes.

You can select a pane by clicking in it. You can also select specific resources within a pane by clicking on the line for the resource. To view or modify the configuration for an item, you can double-click on the item. (You can also use the buttons and menus on this window to access configuration information for specific resources.)

For each item listed, resources that belong to that item are nested within the information for that item. For example, link stations are grouped under the port to which they belong. You can click on the **Expand** button

next to an item to show the resources for that item if they are not currently displayed, or click on the **Contract** button

to hide the resources for an item.

You can perform the following administration tasks from the Node window:

Start or stop a resource
Select the resource and click on the **Start** or **Stop** button. (Alternatively, you can select **Start item** or **Stop item** from the **Selection** menu.)

Add a new resource for an item
Select the item and click on the **New** button (or select **New** from the **Selection** menu). For example, to add a link station for a port, select the port and click on the **New** button.

Delete a resource
Select the resource and click on the **Delete** button (or select **Delete** from the **Selection** menu).

View or modify the configuration for any resource
Select the resource and click on the **Properties** button (or select **Properties** from the **Selection** menu).

Get status information for any resource
Select the resource and click on the **Status** button (or select **Status** from the **Selection** menu).

Copy the configuration for any resource
Select the resource and click on the **Copy** button (or select **Copy** from the **Selection** menu).

In addition, you can choose specific configuration tasks for the node from the **Services** menu, control logging (for the domain) and tracing (for the node) from the **Diagnostics** menu, and view or modify domain resources by selecting one of the items on the **Windows** menu.

Resource Items

The layout of the resources in a window shows the relationships among them.

If an item has one or more child items associated with it, an **Expand** button or **Contract** button appears next to it. An **Expand** button indicates that the associated child items are hidden. You can click on the **Expand** button to show them. A **Contract** button indicates that the child items are shown. You can click on the **Contract** button to hide them. If an item has neither button next to it, the item has no associated child resources.

For example, a link station is associated with a particular port. In the Connectivity pane of the Node window, the link station is displayed below its parent port, along with all other link stations associated with that port. The port is always displayed, but you can choose whether the list of associated link stations is shown or hidden. Similarly, link stations with a list of associated LUs can be expanded to show the LUs, or contracted to hide them.

A parent resource must always be configured before its child resources, and deleting the parent resource causes all its child resources to be deleted.

Tool Bar Buttons

Resource windows include tool bar buttons to make it easy to perform common functions. A tool bar for Communications Server for Linux is shown in Figure 15.

Figure 15. Communications Server for Linux Tool Bar

Not all buttons appear in the tool bars of each resource window. If a button's operation is not valid for the currently selected item (or an operation requires an item to be selected, but none is), the outline of the button is displayed in gray, and the function cannot be selected (the button cannot be pressed). The following buttons can appear on resource windows:

Starts the selected item.

Stops the selected item.

Adds a new resource item. (In the Node window, you add a resource into the selected pane.)

Deletes the selected resources.

Opens the dialog for the selected item to view or modify the item's configuration.

Copies the selected item. Pressing this button opens a dialog whose fields duplicate the configuration of the selected item. Complete the dialog's fields (filling in the new item's name) to add the new resource.

Displays the current status of the selected item.

Many resources, such as ports and link stations, cannot be modified while they are active. You can, however, view an active resource's parameters by selecting the resource and clicking on the **Properties** button to open its dialog, or click on the **Status** button to view detailed status information for the resource.

Resource Dialogs

Resource dialogs show the current configuration information for the resource. A sample dialog for an LU of types 0–3 is shown in Figure 16 on page 48.

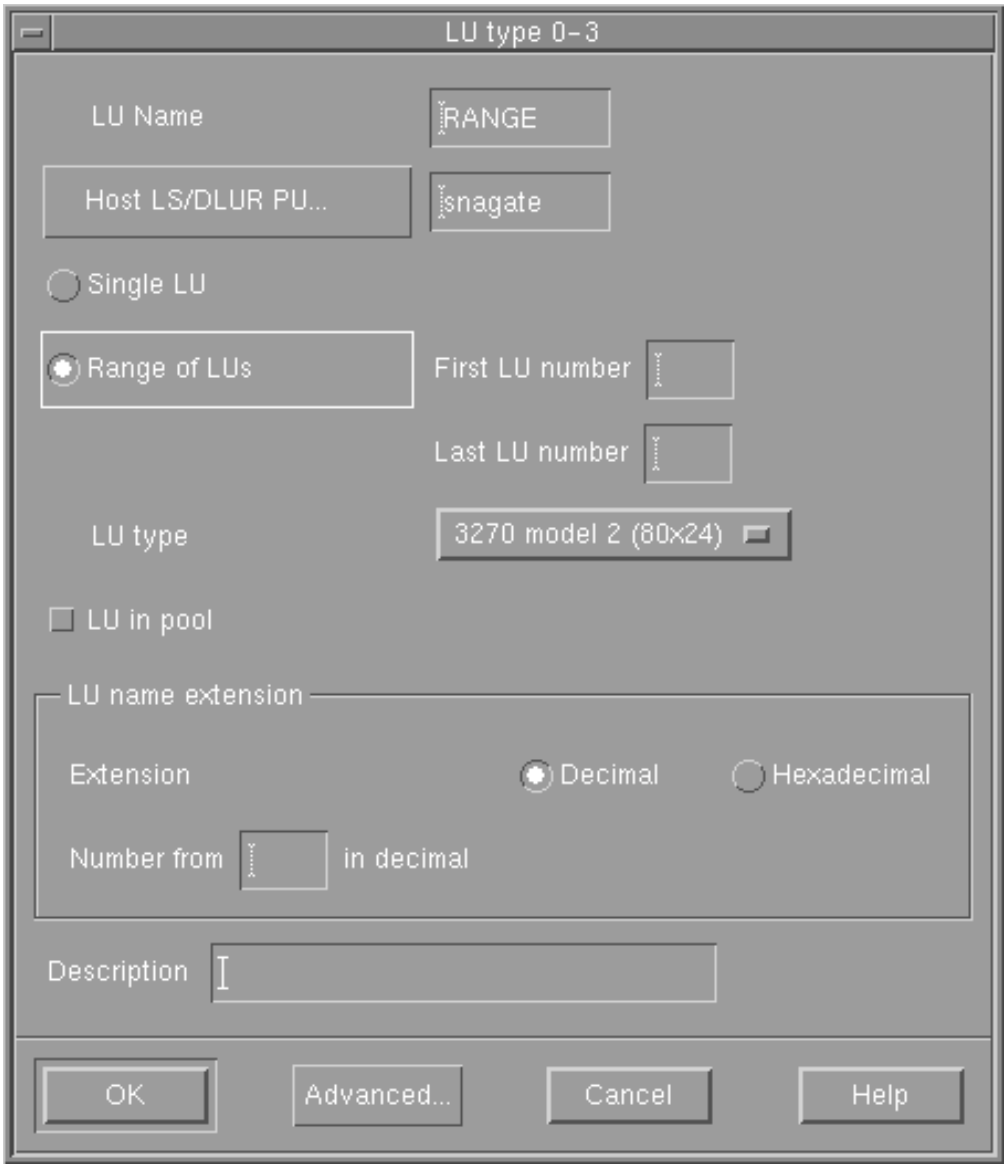

Figure 16. Sample Dialog

Resource dialogs guide you through the configuration process and supply default values whenever possible. For example, when you add a dependent LU, the Motif administration program automatically fills in the *LU number* field with an available LU number on the link station you specify. If you do not supply a required value, the program presents a message pop-up that indicates the information you need to provide.

Most dialogs provide a *Description* field; the information you enter there is displayed on the window where the resource is displayed.

If you are permitted to change the information in a resource dialog (when you are adding a new item or modifying an existing one), the dialog includes **OK** and **Cancel** buttons. Press the **OK** button when you are finished, or the **Cancel** button to exit without changing the configuration for the resource.

If you cannot change the information in a resource dialog (for example if the resource's configuration cannot be modified while it is active), the dialog includes a **Close** button instead of an **OK** button. Click this button when you are finished viewing the information in the dialog.

For context-sensitive help on the dialog, click on the **Help** button.

Note: The basic Motif dialogs expose only the key configuration fields; Communications Server for Linux supplies default values for advanced fields. To access advanced configuration parameters, click on the **Advanced** button. If you decide to adjust advanced parameters, complete the basic dialog before opening the advanced dialog, because that dialog can change depending on the values you enter for basic parameters. For information about advanced configuration fields, see the online help for the Motif administration program.

Status Dialogs

When you select a resource and click on the **Status** button, the Motif administration program shows detailed status information for the resource, as shown in Figure 17.

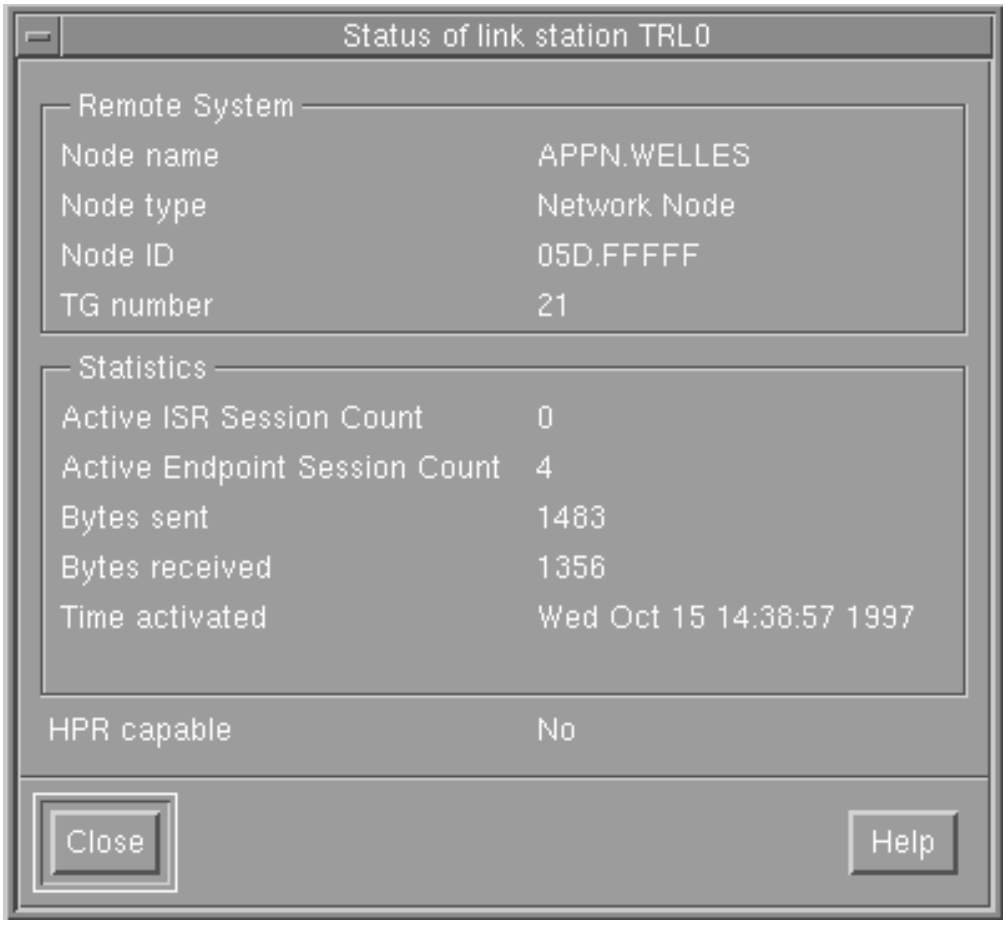

Figure 17. Sample Status Dialog

Status dialogs show information about the current state of the resource. The information is updated dynamically as you view it.

Help Windows

The online help for the Motif administration program provides detailed guidance for each configuration task you need to perform. In particular, task sheets can take you through each step you need to perform in configuring a particular resource. The task sheet for configuring node parameters (always the first step in configuring Communications Server for Linux) is shown in Figure 18.

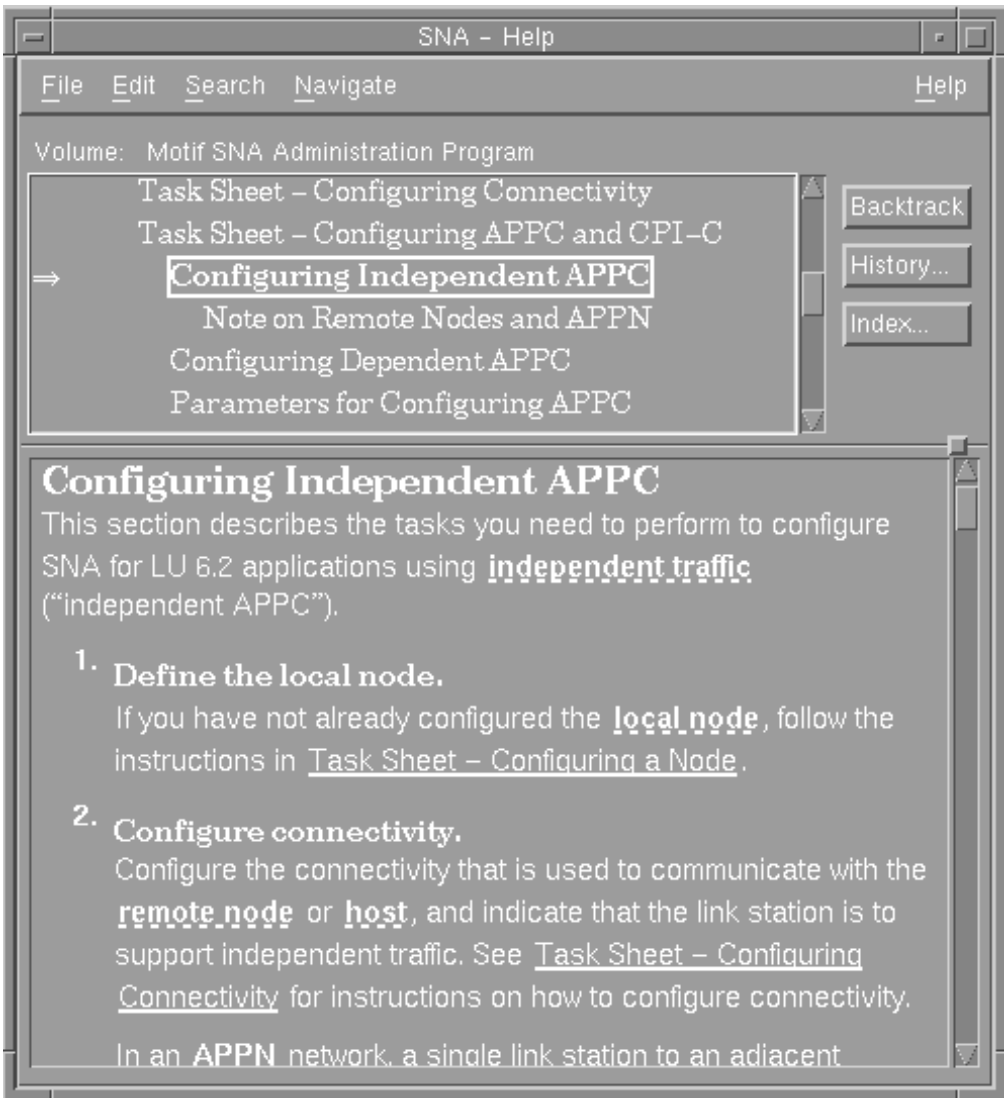

Figure 18. Sample Help Window

Additional help windows are included for each window and dialog, for error messages, and for SNA concepts.

Using the Command-Line Administration Program

Command-line configuration enables you to change all Communications Server for Linux configuration parameters. You can use it to configure any of the resources that are available through the Motif administration program, and can set or change configuration parameters that are not exposed in the Motif program. However, this administration method typically requires that you supply more information than is required for Motif administration. In addition, you must make sure that the

information you provide is valid and consistent with existing resource definitions. (The Motif administration program is recommended because it ensures the data you enter is consistent. In addition, it can infer many configuration values based on menu and dialog choices, and fill in values based on available definitions.)

Most administration commands are used with the **snaadmin** command-line administration program. You can issue **snaadmin** commands in the following form:

snaadmin *command*, *parameter1=value1*, *parameter2=value2*, ...
{*subrecord_name1*}, *sub_param1=sub_value1*,
sub_param2=sub_value2...

You can get help for **snaadmin** command-line administration by using any of the following commands:

- **snaadmin -h** provides basic help for command-line administration and usage information for command-line help.
- **snaadmin -h -d** provides a list of commands that can be supplied to the **snaadmin** program.
- **snaadmin -h** *command* provides help for the named *command*.
- **snaadmin -h -d** *command* provides detailed help for the named *command*, including a list of the configuration parameters that can be specified with the command.

Some commands can be issued from an IBM Remote API Client, provided the command includes the **-n** option to specify a server name. Such a command has the same effect as if it were issued at the named server.

The remainder of this section summarizes administration commands for different types of resources. Some of the types of commands listed are as follows:

status_*
: Provides summary information for types of resources.

define_*
: Creates a new **define_*** record in the configuration file, or replaces an existing record for the same resource with the new definition.

delete_*
: Removes the corresponding **define_*** record from the file.

query_*
: Returns information from the configuration file on the appropriate component, but does not modify the file.

set_* Controls management functions such as tracing and logging parameters.

For complete information about command-line configuration, refer to *Communications Server for Linux Administration Command Reference*.

Chapter 3. Basic Configuration Tasks

This chapter provides an overview of configuration tasks and explains how to configure the Communications Server for Linux node. It also explains how to configure master and backup servers when Communications Server for Linux is used in a client/server environment.

Configuring Client/Server Functions

This section is relevant only if you installed Communications Server for Linux to run in a client/server environment (with multiple Communications Server for Linux nodes in the same network).

Many resources, such as ports and LUs, are configured on an individual node. These are known as "node resources."

Other resources, are common to all nodes; only one definition for the resource is maintained for the entire domain. Such resources are known as "domain resources." Domain resource definitions are stored only on the master server for the domain, and are accessible from all the nodes in the domain.

Note: A standalone Communications Server for Linux system has only one server; that server always acts as the master.

In a client/server environment, a server can be marked as a configuration server; Communications Server for Linux maintains a list of these configuration servers. The first server listed is the master server, and any other servers listed are backup servers. The servers are listed in order, so that the second server listed (the first backup server) takes over if the master server is unavailable, the third server listed (the second backup server) takes over if neither the master nor the first backup server is available, and so on.

When any of the nodes in the domain are active, the first available configuration server in the domain (the first server that can be contacted and has Communications Server for Linux software running) becomes the master server. If the current master becomes unavailable (because it cannot be contacted, perhaps due to a network failure, or because the SNA software running on it is stopped), the next available configuration server in the list becomes the new master.

Communications Server for Linux can run without a master. This happens if none of the servers in the configuration server list can be contacted. If this happens, you can view and configure node resources only on the servers that can be contacted.

Note: You cannot directly indicate which node acts as the master server; the master server is selected based on the order in which nodes are added to the configuration server list. If you wish to move a server to the top of the list, remove all other nodes from the list and then add them again.

You can also use the following administration commands to query, add, and delete configuration servers:

query_sna_net
Lists the servers in the file.

add_backup
Adds a new server to the end of the list.

delete_backup
Removes a server from the list. You can use the **delete_backup** command to delete either the master server (so that the second server listed takes over as master) or a backup server (so that it can no longer act as the master).

Note: You cannot delete a server if it is the only server listed on which the Communications Server for Linux software is running, because in this case there is no other server that can take over as the master server. At least one enabled master server is required in a client/server configuration.

Chapter 10, "Managing Communications Server for Linux Client/Server Systems," on page 123 provides information about advanced Client/Server configuration, including how to move clients and servers into different Communications Server for Linux domains and how to configure the details of client operation.

Configuring the Node

The first step in configuring Communications Server for Linux on a system is to configure the local node. Node configuration provides the basic information that the node needs in order to participate in an APPN network. You must configure the node before you can define connectivity or other resources for the node.

If the node has already been configured, you must stop the node before changing the node configuration.

To configure the node, use one of the following methods:

Motif administration program
Select **Configure node parameters** from the **Services** menu on the Node window.

Command-line administration program
Issue the **define_node** command.

Advanced parameters for node configuration provide control over sessions with undefined partner LUs, reporting of security failures, and limited resource timeouts.

Node Configuration Parameters

You need the following information for node configuration:

APPN support
Level of APPN support for the node:

- If your network is not an APPN network, configure the node as a LEN node.
- To participate in an APPN network in which another node provides session routing services, or to use DLUR only on the local node, configure the node as an end node.
- To provide intermediate routing services in an APPN network, or to provide passthrough DLUR services to downstream nodes, configure the node as a network node.

- To provide session routing services to other nodes in a branch network that are not part of the main APPN backbone network, configure the node as a branch network node.

Control point name
Fully qualified control point name for the local node. Because this name may need to be configured on other nodes in the network, consult with your SNA network planner to determine the name.

When you define the control point, Communications Server for Linux automatically defines a local LU with the same name. That LU can act as a default local LU for the node.

Control point alias
Local alias for the default local LU. Supply this value if the default local LU is used by independent LU 6.2 LUs.

Node ID
Identifier for the PU on the local node. Supply a value only if the node will be used for dependent traffic using the default (control point) LU.

Additional Configuration

After configuring the node, continue with the following configuration tasks:

- Configure connectivity as described in Chapter 4, "Defining Connectivity Components," on page 59.
- Configure node resources (LUs) as described in Chapter 6, "Configuring APPC Communication," on page 79 or Chapter 7, "Configuring User Applications," on page 101.
- Configure applications as described in Chapter 7, "Configuring User Applications," on page 101.

Configuring Logging

Communications Server for Linux writes log messages describing abnormal events (and, optionally, normal events) to log files. When you try to diagnose a problem, the first place to look is in the log files, because the log messages provide information about the cause of the problem and the action you should take.

Communications Server for Linux logs messages for the following categories of event:

Problem
An abnormal event that degrades the system in a way perceptible to a user (such as abnormal termination of a session).

Exception
An abnormal event that degrades the system but that is not immediately perceptible to a user (such as a resource shortage), or an event that does not degrade the system but may indicate the cause of later exceptions or problems (such as receiving an unexpected message from the remote system).

Audit A normal event (such as starting a session).

Communications Server for Linux also maintains a usage log file, which is used to record information about the current and peak usage of Communications Server for Linux resources.

To distinguish between logs relating to normal and error conditions, the different message categories are logged to different files. Problem and exception messages are logged to the error log file; audit messages are logged to the audit log file.

Communications Server for Linux provides a backup mechanism to prevent log files from becoming too large and consuming disk resources. When a log file reaches the maximum permitted size, Communications Server for Linux copies its current contents to a backup file and then clears the log file.

By default, Communications Server for Linux uses the following log files:

Error log file
/var/opt/ibm/sna/sna.err

/var/opt/ibm/sna/bak.err (backup)

Audit log file
/var/opt/ibm/sna/sna.aud

/var/opt/ibm/sna/bak.aud (backup)

Usage log file
/var/opt/ibm/sna/sna.usage

/var/opt/ibm/sna/bak.usage (backup)

You can view the log files using a text editor or other Linux system utilities:

vi View the file in a text editor. This allows you to move through the file forwards or backwards, and to search for particular entries.

pg View a file one page at a time. This utility is simple and easy to use but useful only if the log file is small.

tail View the tail (end) of a file. The end of the file is where the most recent log messages are. Use this utility with the **-f** option to monitor the log file while the system is running.

If you selected succinct rather than verbose logging, you can use the **snahelp** command to determine the cause and action information for a particular message number.

For most purposes, the default settings for logging are sufficient, but you can make the following types of changes:

- Indicate what categories of messages are to be logged.

  Problem messages are always logged and cannot be disabled. Logging is normally disabled for the other two message categories, but you can enable it if necessary.
- Specify the level of detail in logging messages.
- Specify central logging for the domain or local logging for each node
- Change log file names and sizes.

To configure logging, use one of the following methods:

Motif administration program
Select **Logging** from the **Diagnostics** menu on the Node window or the Domain window.

Command-line administration program
Issue one of the following commands:

- **set_central_logging**
- **set_global_log_type**
- **set_log_type**
- **set_log_file**

The Logging dialog in the Motif administration program affects log settings throughout the domain. Using the command line, you can override the domain settings by configuring local log settings on a particular machine.

In addition to providing control over logging, the Motif administration program provides node-level control over tracing. The command-line interface provides greater control over both logging and tracing functions. For more information about logging and tracing, refer to *Communications Server for Linux Diagnostics Guide*.

Chapter 4. Defining Connectivity Components

In order for the Communications Server for Linux node to communicate with other nodes, you must configure connectivity with at least one adjacent node. A connecting link can be configured to carry dependent traffic, independent traffic, or both.

You can have adapter cards for one or more link protocols installed in your computer. Much of the information you need to enter to configure connectivity depends on the link protocol you are using. The remote node must also have an adapter card of the same type you choose, or there must be a bridge or router between the local and remote nodes. For a list of the link protocols supported by Communications Server for Linux, see "Defining DLCs, Ports, and Connection Networks" on page 60.

To configure a link, you need to define a port as described in "Defining DLCs, Ports, and Connection Networks" on page 60. In addition (in most cases), you must configure a link station as described in "Defining Link Stations" on page 64. If LUs on the local node are to communicate with a host using DLUR, you must also define a DLUR PU on the local node as described in "Defining DLUR PUs" on page 71.

When using the Motif administration program, a data link control (DLC) is automatically configured as part of the configuration for the port. In addition, you have the option of defining the port as part of a connection network. When using command-line configuration, this configuration is separate from port configuration.

The information required for link configuration depends on the link protocol, whether your network is an APPN network, and on whether the link is for dependent traffic, independent traffic, or both. In addition, the links that you need to configure depend on what kind of communication you need to support:

LUA If you are going to use LUA, you need to configure a link to the host computer. The link must be configured for dependent traffic, and it must be configured on the host computer as well as on the Communications Server for Linux node, so consult your SNA network planner.

Using CPI-C or APPC
If you are going to use CPI-C or APPC and your network is not an APPN network, you need to configure links to all the adjacent nodes that you want to access. These links must be configured for independent traffic, and they must be configured on the adjacent nodes as well as on the Communications Server for Linux node, so you may need to consult your SNA network planner.

Operating as an APPN Node
If the Communications Server for Linux node is an end node or network node in an APPN network, the number of links that you need to configure can be greatly reduced. You can configure links to one or more adjacent network nodes and access all nodes in the APPN network using these links. If you want to access other adjacent nodes directly, you can configure links to them too—this is not usually necessary, but it can give better performance. If the adjacent nodes are connected by token ring, direct links

can be set up dynamically so you don't need to configure them—just make sure that you configure the network as a connection network when you define the port.

The benefits of APPN networking are always available for and independent APPC, but they do not apply to LUA unless you use DLUR. (DLUR supports communications between a host and dependent LUs on the local node or on downstream nodes in an APPN network.) You can use DLUR only if your host supports DLUS, so you should consult your SNA network planner if you are interested in using DLUR.

Defining DLCs, Ports, and Connection Networks

A port represents the local end of a communications link as a unique access point in the network. Each port is associated with a specific link protocol, which can be any of the following:

- SDLC
- Token ring
- Ethernet
- X.25 or QLLC (qualified logical link control)
- Multipath Channel (MPC) (Communications Server for Linux on System z only)
- Enterprise Extender (HPR/IP)

You can configure more than one port that uses a particular link protocol. In general, a port corresponds to a single physical access point such as an adapter card, but some link protocols (such as token ring) enable you to define multiple ports for a single adapter. The different ports are distinguished by addresses (such as the SAP number).

When you use the Motif administration program to define a port for a particular link protocol, Communications Server for Linux automatically defines a DLC for the port if a DLC of that type has not already been defined. For command-line configuration, you must define the port and DLC using different commands.

In an APPN network using token ringlink protocols, you can also use the SAP Configuration dialog to indicate that the port is part of a connection network.

If you are using SNA gateway, you can define a template that is used to generate definitions for implicit link stations (link stations that are not explicitly configured). Implicit link stations can support downstream LUs. If implicit PU fields are modified while the port is active, the changes affect any implicit link station instances generated after the change.

To configure a port, connection network, and DLC, use one of the following methods:

Motif administration program
Select **Connectivity** and **New port** from the **Services** menu on the Node window.

Command-line administration program
To configure a DLC:

define_*type***_dlc**

To configure a port:

define_*type*_port

In these commands, *type* indicates the link protocol type (`sdlc`, `tr`, `ethernet`, `qllc`, `mpc`, `ip`).

To configure a connection network:

define_cn

Advanced port configuration parameters provide control over BTU size, the number of active links permitted, generation of implicit downstream LUs, and settings for dynamic link stations.

DLC, Connection Network, and Port Configuration Parameters

The following parameters are required for port configuration. (When you use the Motif administration program, port configuration also supplies information about the DLC and enables you to assign a port to a connection network.)

SNA port name
The locally known name of the port.

Adapter card number
This field is not used for Enterprise Extender ports.

A number that identifies the adapter card to use, if you have more than one card of the same type in this computer.

Port number
This field is not used for Enterprise Extender ports.

The number of the port to be used, if the adapter card can support more than one port. The range of valid port numbers is from `0` to the number of ports supported by the adapter card minus one. For the first port on the adapter card, enter `0`.

This field applies only if the adapter card can support more than one port.

Initially active
Whether to activate the port automatically when the node is started. This setting enables link stations that use the port to be activated in response to requests from adjacent nodes or on demand by the local node. (Activating the port does not activate any link stations; link stations are activated separately.)

The following sections describe additional port parameters that are specific to the link type. No additional port parameters are required for QLLC.

Additional Port Parameters for SDLC

Line details
The following parameters describe the type of SDLC connection:

Type Select one of the following values:

`Leased Line`
A dedicated line is used for the SDLC link between this computer and the remote system.

`Switched incoming`
The standard telephone network is used for incoming calls.

For a nonprimary port (as indicated by the *Link role* field), you also need to configure the poll address (for outgoing calls, that address is configured on the link station). The poll address is a one-byte address (`C1` by default) that needs to match the poll address configured at the remote link station. When active, the port responds to frames sent with this poll address.

For a primary port, you do not need to configure a poll address; the port uses the poll address specified by the remote link station on the incoming call. For other types of ports, the poll address is configured on each link station.

`Switched outgoing`
The standard telephone network is used for outgoing calls.

Link role
Select a value that describes the role of the local node for link stations defined on this port. In SDLC communication, one end manages the link and is called the primary link station. The other end is the secondary link station.

Use one of the following values for this field:

`Secondary`
The other end of the link is to be the controller and the remote system is configured to be primary. This is nearly always the case if you are configuring a link to a host system.

`Primary`
This port is to act as the SDLC controller of the link, and the remote system is configured to be secondary.

`Negotiable`
For maximum flexibility, this setting enables the two ends to negotiate which end performs the primary role. Choose this value if you do not know which role is configured for the remote system.

You can use this setting for a peer link, but be aware that negotiating the role causes a short delay when the link is activated.

`Primary Multi-drop`
The link is leased and this port is to act as controller of a multi-drop link to several secondary nodes.

Use this setting when you want to configure several link stations from the local node to different remote nodes (for example, for links to downstream nodes). Each of these other nodes must be configured as secondary, and you must be using a leased line.

`Secondary Multi-PU`
The local port is one of the secondary stations on a multi-drop link controlled by the port on the remote system.

Consult your SNA network planner if you do not know how to configure any of these parameters.

Additional Port Parameters for Token Ring and Ethernet

Local SAP number
: The address of the SAP, usually `04`for Intel and OSA2 adapters.. Use a different value only if you need to use more than one SAP on the card. For an OSA-Express adapter, the local SAP number must match that defined in the OSA/SF for the I/O device addresses that correspond to the ethX interface on this Linux image.

  The SAP number must be a multiple of 4.

  If you do not know what value to enter for this field, contact your SNA network planner.

Define on connection network
: Whether the SAP is to access the LAN as a connection network. Defining a connection network enables links between nodes on the connection network to be started dynamically, without prior configuration.

  This field applies only if the local node is not a LEN node, because LEN nodes cannot use connection networks.

CN name
: The name of the connection network. You do not need to enter the CN name unless you specified the *Define on connection network* option to define the SAP on a connection network. The CN name is used as the name of a virtual routing node in order to establish links between the nodes on the connection network.

  Specify the same CN name on all nodes on the connection network.

Ethernet type
: This field applies only to Ethernet links.

  Whether the network is a standard Ethernet network or an IEEE 802.3 network.

Additional Port Parameters for Enterprise Extender (HPR/IP)

Local IP interface
: This is an optional field. It allows you to specify the local IP network interface to be used for the port, if you have access to multiple IP networks. If you have access to only one IP network, you can leave this field blank.

  If you need to specify the interface, you can use any of the following.

  - An IPv4 dotted-decimal address (such as `193.1.11.100`).
  - An IPv6 colon-hexadecimal address (such as `2001:0db8:0000:0000:0000:0000:1428:57ab` or `2001:db8::1428:57ab`).
  - An interface identifier (such as `eth0` or `en0`).

Protocol
: Whether links on this port use IPv4 or IPv6 addresses.

Define on connection network
: Whether the adapter card is to access the LAN as a connection network. Defining a connection network enables links between nodes on the connection network to be started dynamically, without prior configuration.

  This field applies only if the local node is not a LEN node, because LEN nodes cannot use connection networks.

CN name
The name of the connection network. You do not need to enter the CN name unless you specified the *Define on connection network* option to define the port on a connection network. The CN name is used as the name of a virtual routing node in order to establish links between the nodes on the connection network.

Specify the same CN name on all nodes on the connection network.

Additional Port Parameters for Implicit Links

Maximum active template instances
Specify the maximum number of link station instances to be generated from the template.

Configure downstream LUs for implicit PU access
Whether to configure downstream LUs that use this PU (see "Configuring SNA Gateway" on page 111).

HPR supported on implicit links
Whether to support High-Performance Routing on implicit link stations.

Link level error recovery on implicit links
Whether to send HPR traffic on implicit links using link-level error recovery.

Additional Configuration

After performing the port configuration, continue with the following configuration tasks:

- To define a link station on a port you have configured, see "Defining Link Stations."
- To define a DLUR PU, see "Defining DLUR PUs" on page 71.
- To support APPC communication, see Chapter 6, "Configuring APPC Communication," on page 79.

Defining Link Stations

To communicate with other nodes in an SNA network, you must configure the characteristics of a link station (LS) to an adjacent node in the SNA network. Before you can define a link station, you must define a port for the adapter (and link protocol) you are using. Most of the information needed to configure a link station is the same, whatever protocol is being used.

A link station represents the logical path through the SNA network between the Communications Server for Linux local node and a remote computer. The remote computer can be any of the following:

- A host computer, on which Communications Server for Linux accesses a host program using 3270 or LUA communications (or uses APPC or CPI-C for program-to-program communications)
- A peer computer, with Communications Server for Linux and the remote computer communicating as equal partners (the typical arrangement in an APPN network)
- A downstream computer that uses the Communications Server for Linux SNA gateway feature or DLUR feature in order to access a host

A link station is associated with a specific port; you can define one or more link stations on each port.

Each link station that supports dependent traffic has an associated PU (physical unit). Because PUs are associated with link stations, Communications Server for Linux does not treat them as separate resources; they are configured as part of link station configuration, and are started and stopped as part of starting and stopping link stations. Link stations are shown in the connectivity section of the Node window; PUs are not shown in any window.

Note: In most circumstances, you need to add a link station to the port. However, if you want to use a dynamically created link station for downstream SNA gateway or for APPC traffic only, for situations in which the link is always activated from the remote node, you do not need to explicitly configure one.

If a remote node attempts to connect to the local node, but no link station is defined that matches the address specified on the incoming call, Communications Server for Linux can define one implicitly if a suitable port has been defined on the local node. This dynamically created link station appears in the connectivity section of the Node window for the duration of the connection.

To configure a link station, use one of the following methods:

Motif administration program
Select **Connectivity** and **New link station** from the **Services** menu on the Node window.

Command-line administration program
Issue the following command:

define_*type*_**ls**

In this command, *type* indicates the link protocol type (`sdlc`, `tr`, `ethernet`, `qllc`, `mpc`, `ip`).

Advanced parameters for link stations provide additional control over transmission characteristics, XID exchange, optional link facilities, compression for LU 0–3 sessions using the link, and reactivation procedures.

Link Station Configuration Parameters

In Motif, the Link Station Configuration dialog contains the following sections, each containing different categories of configuration parameters:

Link station
Use this area of the dialog to provide information that is required for all link stations, whether they support LU traffic for dependent LUs, independent LUs, or both. For descriptions of the parameters in this section, see "Common Link Station Parameters" on page 66.

Independent LU traffic
Provide this information only if you are using the link station for independent traffic. For descriptions of the parameters in this section, see "Parameters for Independent LU Traffic" on page 68.

Dependent LU traffic
Provide this information only if you are using the link station for dependent traffic. For descriptions of the parameters in this section, see "Parameters for Dependent LU Traffic" on page 69.

Common Link Station Parameters

The following parameters are required for all link stations, whether they support dependent traffic, independent traffic, or both.

For more information about the parameters on this dialog, refer to the online help or to *Communications Server for Linux Administration Command Reference*.

Name A name to identify the link station locally.

SNA port name
The port that is to be used to access the adjacent node.

Activation
Method used to activate the link station. Specify one of the following methods:

By administrator
The link station is activated only on the request of a local System Administrator.

On node startup
The link station is started automatically when the node starts up.

On demand
The link station is started automatically when required to provide connectivity for an application.

Link stations are activated separately from ports, so the link station must be activated even if the port is already active. Activating the port does not itself activate any link stations, and configuring the port to be initially active does not mean that any of its link stations are activated automatically when the node starts up. However, activating a port does make it possible to activate link stations. A link station cannot be activated unless the ports are active on both the local node and the adjacent node.

If the link is one for which you are charged for usage, avoid activating the link unnecessarily, in order to keep the cost down.

If you are not sure how to set this field, consult your SNA network planner.

LU traffic
The type of LU traffic to flow over the link. This choice determines what other parameters are needed for link definition.

This parameter is not used for an Enterprise Extender (HPR/IP) link, because this link type supports only independent traffic.

Any The link station can be used for both independent and dependent LU traffic. For this option, you must supply values for the fields described in "Parameters for Independent LU Traffic" on page 68 and "Parameters for Dependent LU Traffic" on page 69, in addition to those described in this section.

Independent only
The link station can be used only for independent LU traffic. For this option, you must supply values for the fields described in "Parameters for Independent LU Traffic" on page 68, in addition to those described in this section.

Dependent only
The link station can be used only for dependent LU traffic. For this

option, you must supply values for the fields described in "Parameters for Dependent LU Traffic" on page 69, in addition to those described in this section.

You also need to provide addressing information for contacting the adjacent node. The type of addressing information needed depends on the DLC type of the port. If you do not supply an address for the remote node, the link station acts as a nonselective listening link station, accepting incoming calls from any remote node.

Additional Link Station Parameters for SDLC:

Poll address

The poll address of the remote station. Specify the address as a two-digit (one-byte) hex value, typically starting at `C1`. A primary link station polls the remote station using this value. A secondary link station responds to polling with this value. The poll address is entered differently depending on the link role:

- If the link is a point-to-point link (not multi-drop), the address `C1` is normally used.
- If the parent port for this link is switched incoming, the poll address is configured on the port and cannot be configured independently for each link station.
- If you are configuring a primary switched outgoing link station, and you do not know the poll address of the remote secondary with which you wish to communicate, you can specify a poll address of `0xFF` on the primary. This value enables the node to accept responses from a secondary, regardless of the poll address it has configured. `0xFF` is not a valid address for a nonprimary link or a link that is not switched outgoing.
- If you are using a multi-drop configuration, all the secondary link stations that communicate with the same primary must have different poll addresses.

The poll addresses at both ends of the link must match. Contact your SNA network planner if you do not know the address configured at the remote system.

On a VTAM host, the poll address is configured as the *ADDR*= parameter in the VTAM PU definition.

On an AS/400 system, the poll address is the *STNADR* parameter of the Line Description.

Additional Link Station Parameters for Token Ring and Ethernet:

MAC address

The MAC address of the remote station, entered as a series of hexadecimal digits. The MAC address uniquely identifies the adapter card on the remote system.

If you do not know what value to use, consult your SNA network planner.

If the remote end of this link is a VTAM host, you can find its MAC address in the *MACADDR*= parameter of the VTAM Port definition.

If you are configuring a link to an AS/400 system, the MAC address is the *ADPTADR* parameter in the Line Description.

SAP number

The SAP number of the port on the remote computer. The SAP number

distinguishes between different links using the same adapter card. This is a hex number, normally `04`. It must be a multiple of 4.

If you do not know what value to use, consult your SNA network planner.

If the remote end of this link is a VTAM host, the SAP number is the *SAPADDR*= parameter of the VTAM PU definition.

If you are configuring a link to an AS/400 system, the SAP number is the *SSAP* parameter in the Line Description.

Additional Link Station Parameters for X.25 (QLLC):

Remote X.25 address
If the link is a switched virtual circuit, enter the DTE address of the remote DTE as a series of hexadecimal digits.

If the link is a permanent virtual circuit, enter the channel ID that identifies the virtual circuit the link station is to use. Channel IDs are numbered from 1 up to a maximum of 1024. If you have only one permanent virtual circuit, its channel ID is likely to be 1.

Additional Link Station Parameters for MPC:

MPC group
The MPC (MultiPath Channel) group name specified in the MPC driver configuration to identify a particular channel.

Additional Link Station Parameters for Enterprise Extender (HPR/IP):

Remote IP host name
Remote host name of the destination node for this link. The hostname can be any of the following; the *protocol* parameter on the port that this link uses determines whether the address should be in IPv4 or IPv6 format.

- An IPv4 dotted-decimal address (such as `193.1.11.100`).
- An IPv6 colon-hexadecimal address (such as `2001:0db8:0000:0000:0000:0000:1428:57ab` or `2001:db8::1428:57ab`).
- A name (such as `newbox.this.co.uk`).
- An alias (such as `newbox`).

If you specify a name or alias, the Linux system must be able to resolve this to a fully qualified name (either using the local TCP/IP configuration or using a Domain Name server).

Parameters for Independent LU Traffic

You need the following information to configure this link station for use by independent LUs (LUs of type 6.2 for use by APPC, 5250, or CPI-C applications):

Remote node name
The fully qualified CP name of the remote node.

If the remote system is a VTAM host, you can find the network name (the first eight characters of the fully qualified name) in the *NETID* parameter of the VTAM start list. The last eight characters are in the *SSCPNAME* parameter of the VTAM start list.

Note: If you enter the name of a new remote node, you can add a definition for the remote node to enable you to define partner LUs

on the new remote node. If the local node is a LEN node, you do not need to specify the remote node type, and the *Remote node type* field does not apply.)

To define a new remote node in this way, specify the remote node type as a value other than `Discover`, as well as specifying the remote node name.

Alternatively, you can specify `Discover` and leave the remote node name empty. This means that any adjacent node can use the link station. The `Discover` option is not available if the local node is a LEN node.

Remote node type
The level of APPN support on the remote node that is accessed through this link station (only applicable if the local node is an end node or network node).

If you do not know whether the remote node is a LEN node or end node or whether it is a network node, you can choose `Discover`. Discovering the level of APPN support on the remote node can delay link activation slightly, so if you do know the type it is better to specify it. This also helps to ensure network configuration consistency.

You cannot choose `Discover` if the link station is activated on demand.

If the local node is a LEN node, this field does not apply.

Branch link type
The type of link to the remote node that is accessed through this link station (only applicable if the local node is a branch network node).

If the remote node is a network node within the main APPN backbone, choose `Uplink (to backbone)`. If the remote node is an end node within the branch, choose `Downlink (within branch)`.

If the remote node is configured to be a network node, the branch link type is automatically set to `Uplink (to backbone)` and you cannot change it.

Parameters for Dependent LU Traffic

These parameters do not apply to an Enterprise Extender (HPR/IP) link, because this link type supports only independent traffic.

Configuring a link station for dependent LU traffic automatically creates an appropriate PU with the same name as the link station.

You need the following information to configure a link station for use by dependent LUs (LUs of type 0–3 for 3270 or LUA applications):

Local node ID
A value to identify the local node in the SNA network.

You can usually use the same node ID (the default value) for all the links on the same node. However, if you need more than 255 dependent LUs to access a specific host, you must configure multiple link stations to the host, each with up to 255 dependent LUs, and each with a different local node ID.

To ensure that the remote node is configured to recognize the local node ID, contact your SNA network planner.

In a VTAM configuration, the first three digits should match the *IDBLK* parameter in the PU definition, and the last five should match the *IDNUM* parameter.

On an AS/400 system, the node ID is configured in the *EXCHID* parameter.

Remote node ID
The node ID for the remote link station (optional; only applicable if you need to restrict access to this link station). If you specify the remote node ID, the link is activated only if the node ID of the remote node matches the value specified in this definition. This can be useful if you have several link stations configured on a switched port, because it enables the link stations to be distinguished when they are activated by the remote nodes. Link stations can also be distinguished by the CP name of the remote node, but for remote nodes that do not send their CP name when activating a link, the remote node ID must be used instead.

If you do not specify the remote node ID, the node ID of the remote node is not checked when the link is activated.

Remote node role
The role of the remote (adjacent) node:

Host The link station supports dependent LUs (such as 3270 LUs) that are used for sessions with a host computer (the most common case). If the link is to a node that provides host connectivity using SNA gateway or DLUR, the adjacent node role should still be set to `Host`, even though the link is not directly to a host computer.

Downstream (SNA gateway)
The link station is to a downstream node that will communicate with a host using the SNA gateway capabilities of the local node (to the host, the LUs on the downstream node appear to reside on the local node).

Downstream (DLUR)
The link station is to a downstream node that will communicate with a host using the DLUR capabilities of the local node. (To the host, the LUs on the downstream node appear to reside on the local node.)

Such links can be used only if the local node is an APPN network node.

Downstream PU name
The PU name associated with the downstream node. This value must match the PU name that is configured for the downstream node on the host computer. If you do not know the value to use for this name, consult your SNA network planner.

This field applies only if you specified that this link station is to a downstream PU that will communicate with a host using the DLUR capabilities of the local node. You can indicate this by specifying `Downstream (DLUR)` for the *Remote node role* field.

For more information, see "Defining DLUR PUs" on page 71.

Upstream DLUS name
The fully qualified LU name of the host LU that supports DLUS (the LU server that the downstream PU is to access). If you do not know the value to use for this name, consult your SNA network planner.

This field applies only if you specified that this link station is to a downstream PU that will communicate with a host using the DLUR capabilities of the local node. You can indicate this by specifying `Downstream (DLUR)` for the *Remote node role* field.

Additional Configuration

After performing the link station configuration, continue with the following configuration tasks:

- To define a DLUR PU, see "Defining DLUR PUs."
- To configure passthrough services, see Chapter 8, "Configuring Passthrough Services," on page 103.
- To support specific user applications, see Chapter 7, "Configuring User Applications," on page 101.
- To support APPC communication, see Chapter 6, "Configuring APPC Communication," on page 79.

Defining DLUR PUs

Normally, a dependent LU session requires a direct communications link to the host computer. If many nodes (including a host node) are connected together in an APPN network, some of them may not have a direct connection to the host, but only an indirect connection through another node. It is not possible to establish dependent LU sessions to the host from LUs in these indirectly connected nodes.

Dependent LU requester (DLUR) is an APPN feature designed to overcome this limitation.

This section explains how to configure a DLUR PU that provides connectivity to a host computer. Configuring a DLUR PU enables the local node to provide DLUR services.

DLUR on an APPN node (such as a node running Communications Server for Linux) works in conjunction with dependent LU server (DLUS) at the host, to route sessions from dependent LUs on the DLUR node across the APPN network to the DLUS host. The route to the host can span multiple nodes and can take advantage of APPN's network management, dynamic resource location, and route calculation facilities. DLUR must be available on the node where the LUs are located, and DLUS must be available on the host node, but DLUR is not required on any intermediate nodes in the session route.

If the Communications Server for Linux DLUR node is a network node, it can also provide passthrough DLUR facilities for dependent LUs on downstream computers connected to the Communications Server for Linux node. These LUs can use DLUR on the Communications Server for Linux node to access the host across the network, in the same way as for LUs internal to the node.

To provide passthrough DLUR services to a downstream node, you must first configure (on the local node) the PU name associated with the downstream node. This value must match the PU name that is configured for the downstream node on the host computer.

To configure a DLUR PU, use one of the following methods:

Motif administration program
Select **Connectivity** and **New DLUR PU** from the **Services** menu on the Node window.

Command-line administration program
Issue the following command:

define_internal_pu

DLUR PU Configuration Parameters

The following parameters are required for DLUR PU configuration:

PU Name
For each DLUR PU on the local node, specify a PU name. This name should match the PU name configured on the host. (Communications Server for Linux sends both the PU name and PU ID to the host to identify the PU. The host normally identifies the PU by its PU name, or by the PU ID if it cannot find a matching PU name.)

DLUS Name
The fully qualified LU name of the host LU that supports DLUS.

In order to use DLUR, the DLUR component of Communications Server for Linux has to establish an LU-LU session with the DLUS on the host.

Contact your SNA network planner to determine the name of the host LU.

Backup DLUS Name
This parameter is optional. The fully qualified LU name of a backup host LU that can be used if the one specified by *DLUS Name* is unavailable.

Contact your SNA network planner to determine the name of the host LU.

PU ID The PU ID of the PU on the local node that supports connectivity to the host. The PU ID comprises two hexadecimal strings, one of three digits (known as the block number), and one of 5 digits.

Each dependent LU is associated with a PU. Both the PU and the LU are configured on the host computer. For each PU, you need to define a DLUR PU on the Communications Server for Linux node. The PU ID must match the PU ID configured at the host for this PU.

In many cases the PU ID is the same as the node ID, so the node ID is the default. However, if you need more than 255 dependent LUs to access a specific host, you need to configure multiple DLUR PUs, each with up to 255 dependent LUs, and each with a different PU ID.

If you are not sure how to set this field, consult your SNA network planner.

In a VTAM configuration, the first three digits should match the *IDBLK* parameter in the PU definition, and the last five digits should match the *IDNUM* setting.

Initially active
Whether the DLUR PU is to be activated automatically when the node is started. If you do not set this option, the DLUR PU must be started manually.

Compression supported
Whether data compression is supported for LU 0–3 sessions using this PU.

If you set this option, compression will be used if the host requests it. If you do not set this option, compression will not be used.

Retry contacting DLUS indefinitely
Whether Communications Server for Linux retries the attempt to contact the DLUS if it fails on the first attempt. If you set this option, Communications Server for Linux will retry indefinitely if the first attempt fails. If you do not set this option, it will only retry once.

Parameters for Passthrough DLUR for Downstream Nodes

You need the following information in order to configure Communications Server for Linux to use passthrough DLUR to transport traffic between dependent LUs on downstream nodes and a host:

Downstream PU name
The PU name associated with the downstream node. The PU name must match the PU name configured on the host computer.

A downstream node can support multiple PUs. In this case, each downstream PU is associated with a different link, so you need to configure multiple links between the Communications Server for Linux DLUR node and the downstream node, and you need to know the downstream PU name for each link.

Consult your SNA network planner to find out the PU names associated with the downstream nodes.

DLUS name
The fully qualified LU name of the host LU that supports the DLUS. In order to use DLUR, the DLUR component of Communications Server for Linux has to establish an LU-LU session with the DLUS on the host.

Consult your SNA network planner to find out the LU name for the host DLUS server.

Additional Configuration

After configuring DLUR, continue with the following configuration tasks:

- To configure LUs for DLUR, see "Configuring DLUR" on page 113.
- To configure other passthrough services, see Chapter 8, "Configuring Passthrough Services," on page 103.
- To support specific user applications, see Chapter 7, "Configuring User Applications," on page 101.
- To support APPC communication, see Chapter 6, "Configuring APPC Communication," on page 79.

Chapter 5. Configuring Dependent LUs

This chapter provides instructions for configuring LUs and LU pools to support user applications that use 3270, TN3270and LUA communications. To use these, you must configure dependent LUs.

Before you can configure the resources described in this chapter, you must perform the following configuration:

- Configure the node as described in "Configuring the Node" on page 54.
- Configure connectivity as described in Chapter 4, "Defining Connectivity Components," on page 59. For 3270, TN3270and LUA, you must configure the link to support dependent LU traffic.

  You do not need to configure a direct link to the host if you are using upstream SNA gateway or DLUR. For more information, see "Configuring SNA Gateway" on page 111 and "Defining DLUR PUs" on page 71.

Defining LU Types 0–3

You must configure dependent LUs of types 0–3 to support communication with a host system. You can use the information in this section to define an LU to support 3270 or LUA. You can also define a range of LUs, to configure multiple LUs of the same type in a single operation.

To configure an LU of types 0–3, use one of the following methods:

Motif administration program
Select one of the following from the **Services** menu on the Node window.

- **3270** and either **New 3270 display LU** or **New 3270 printer LU**
- **LUA** and **New LUA LU**
- **TN server** and **New host LU**

Command-line administration program
Issue one of the following commands:

define_lu_0_to_3

define_lu_0_to_3_range

You can use the advanced dialog to restrict access to a specific SSCP, or to specify an inactivity timeout.

LU Types 0–3 Configuration Parameters

The following parameters are required for LU types 0–3 configuration:

LU name
An LU name of 1–8 characters (for a single LU) or a base name of 1–5 characters (for a range of LUs, a prefix is added to the base name to form all of the names for the LUs that are defined).

The LU name is used only locally; it does not need to correspond to a name defined on the host.

Host LS/DLUR PU
The link station that provides the link to the host. The LU definition belongs to the link station you select. (If the dependent LU resides on a node that supports DLUR, this field identifies the DLUR PU that provides connectivity to the host.)

LU numbers
An LU number or range of LU numbers. LU numbers can be from 1–255.

The LU numbers must correspond to those in the host VTAM configuration. If you do not know what numbers are configured on the host, consult your SNA network planner.

LU type
One of the following LU types (depending on the type of LU you are configuring):

- If you are using this LU with TN Server and DDDLU at the host, the LU may not be configured at the host. In this case, the LU type you specify here is used to define the LU on the host dynamically. Specify `Unrestricted` (`unknown` for command-line configuration) if you want the LU model type to be defined to match the type requested by the downstream TN3270 client.
- For a 3270 display LU, specify the appropriate model based on the screen size:
  - `3270 model 2 (80x24)`
  - `3270 model 3 (80x32)`
  - `3270 model 4 (80x43)`
  - `3270 model 5 (132x27)`
- For a printer LU, specify one of the following:
  - `3270 printer`
  - `SCS printer`
- If you do not know the LU type, if the LU is used to support SNA gateway from the local node to the host (an upstream LU), or if the LU is for an LUA application, specify `Unrestricted` (`unknown` for command-line configuration).

The LU type should match the configuration of the LU at the host. If necessary, the LU type configured at the host takes precedence.

Depending on the value you specify, Communications Server for Linux sends one of the following strings to the host in the DDDLU NMVT, to match the values used in the standard VTAM tables:

`3270002` for `3270 model 2`
`3270003` for `3270 model 3`
`3270004` for `3270 model 4`
`3270005` for `3270 model 5`
`3270DSC` for `3270 printer`
`3270SCS` for `SCS printer`
`3270000` for `RJE workstation`
`327000`*n* for `Unrestricted` with a TN3270 client, where *n* is the model number (2–5) provided by the client
`327000@` for `Unrestricted` with an LUA client

LU in pool
Whether the LU is assigned to an LU pool.

Note: If you assign this LU to an LU pool, and assign a user's session to this LU, the user's session uses this LU if it is available; otherwise it uses any free LU from the pool, as though you had assigned it to the LU pool instead of the specific LU. If you want the user to use only a specified LU, so that the user's session cannot be established if the LU is already in use, ensure that the LU is not in a pool.

Pool name
The name of the LU pool.

Additional Configuration

After performing the LU type 0–3 configuration, continue with the following configuration tasks:

- To use a pool of dependent LUs for a 3270 display, for TN3270 or for LUA, define the LU pool as described in "Defining LU Pools."
- For TN3270, define TN3270 client access records as described in "Configuring TN Server" on page 103.

Defining LU Pools

For 3270, TN3270 and LUA, you can define LU pools to simplify user configuration and provide greater flexibility in establishing host sessions.

Note: You can assign a user's session either to a specific LU or to an LU pool.

- If you assign the user's session to a specific LU that is in a pool, the session uses this LU if it is available; otherwise it uses any free LU from the pool, as though you had assigned it to the LU pool instead of the specific LU.
- If you want the user to use only a specified LU, so that the user's session cannot be established if the LU is already in use, ensure that the LU is not in a pool.

LU pools can even span multiple Communications Server for Linux servers—just define LU pools with identical names on the different servers. If a server fails or is taken out of service, clients that use the LU pool can then use a different server. Using LU pools also simplifies client configuration and makes it easy to increase capacity by adding another server or by adding LUs on an existing server.

You can view all of the LU pools for the Communications Server for Linux domain using the LU Pools window. This window lists the LU pools configured in the system, and enables you to select LUs to add to an LU pool. The individual LUs in an LU pool are listed below the LU pool.

An LU is identified as follows:

- 3270 display LU
- Unrestricted LU
- SCS Printer
- 3270 Printer

Do not mix LUs of different types in the same pool (for example, do not put display and printer LUs into the same pool). It is unlikely that you will need a pool of printer LUs unless you are supporting TN3270E clients.

To configure an LU pool, use one of the following methods:

Motif administration program
Select **LU Pools** from the **Windows** menu on the Node window, then choose **New** to add a pool.

Command-line administration program
Issue the following command:

define_lu_pool

LU Pool Configuration Parameters

The following parameters are required for LU pool configuration:

Name A name to identify the LU pool. This field applies only when you are adding a new LU pool. You cannot change the name of an existing pool.

Assigned LUs
LUs to be assigned to the pool. An LU can only be a member of one pool.

Chapter 6. Configuring APPC Communication

APPC applications, 5250 emulation programs, and CPI-C applications all require that you configure APPC first. An APPC application uses the node's LU type 6.2 resources to communicate with another APPC or CPI-C application on a host or peer computer, using a specified mode.

If the applications use CPI-C, you may need to do additional CPI-C configuration after configuring APPC. A CPI-C application uses the node's LU type 6.2 and mode resources to communicate with another APPC or CPI-C application on a host or peer computer. You define the same resources for a CPI-C application as for an APPC application. In addition, if the TP on the Communications Server for Linux computer is the invoking TP (the TP that starts the conversation), you may need to define one or more side information entries for it, as described in "Defining CPI-C Side Information" on page 96. Each of these entries provides information on a partner TP, the LU and mode resources used to access it, and any security information required.

The configuration steps for APPC depend on whether the LU 6.2 traffic is dependent or independent. Unless the remote node is a host, you must use independent traffic. If the remote node is a host, you can use either dependent or independent traffic.

Before you can configure APPC communication, you must perform the following configuration:

- Configure the node as described in "Configuring the Node" on page 54.
- Configure connectivity as described in Chapter 4, "Defining Connectivity Components," on page 59.

  Note: In an APPN network, a single link station to an adjacent network node can be used to communicate with any remote node in the network, so you do not need to configure a separate link station to each remote node.

In many cases, APPC applications can use the control point LU on both the local and remote nodes, and a standard mode. In this case, your configuration is ready for APPC without any further configuration.

The following steps can be used to configure APPC communication on the local node. Depending on the types of the local and remote nodes, and on your application, you may not need to perform these steps.

1. Define a local LU as described in "Defining Local LUs" on page 80.
2. Define a remote node as described in "Defining Remote Nodes" on page 81.
3. Define a partner LU as described in "Defining Partner LUs" on page 83.
4. Define an invokable TP as described in "Defining TPs" on page 86.
5. Define a mode as described in "Defining Modes and Classes of Service" on page 92.
6. Define CPI-C side information as described in "Defining CPI-C Side Information" on page 96.
7. Define APPC security as described in "Configuring APPC Security" on page 98.

8. To configure 5250 communication, see Chapter 7, "Configuring User Applications," on page 101.

Defining Local LUs

In many cases, applications can use the local node's control point LU, which is automatically defined when you configure the node. This is the default LU—if your application does not specify a particular LU, it can use this one. If the application uses the default LU, you do not need to define a local LU. Check the documentation for your APPC application, or contact the application programmer.

If you are configuring dependent LUs of type 6.2 for use with APPC or CPI-C applications, you may wish to define them as members of the default pool. An application that does not specify a particular local LU is assigned an unused LU from the pool of LUs defined as default LUs.

You can define dependent LU 6.2s as default LUs (and you can define default LUs on more than node). An application requesting a default LU can be assigned to any of these LUs as available; the LU does not have to be on the same computer as the application. However, if you are defining partner LUs for the applications, the partner LUs must be defined on all nodes where default LUs are defined, so that the application can contact the correct partner LU using any of the default local LUs defined on any node.

Independent APPC and 5250 use independent LUs. Each LU-LU session involves a local LU and a partner LU. For the local LU, you can use the predefined default LU associated with the node control point, or you can configure new local LUs. The partner LU need not be configured at all if the Communications Server for Linux node is an end node or network node in an APPN network, because APPN can locate partner LUs dynamically. However, you do have to configure the partner LU if your network is not an APPN network or if the node is a LEN node. In this case, you must configure the remote node where the partner LU resides, then define the partner LU on the remote node. (If the partner LU is the default LU on the remote node, you do not need to define it explicitly because it is added automatically when you define the remote node.)

To configure an APPC local LU, use one of the following methods:

Motif administration program
Select **APPC** and either **New independent local LU** or **New dependent local LU** from the **Services** menu on the Node window.

Command-line administration program
Issue the following command:

define_local_lu

You can use the advanced dialog to specify sync point support, attach routing characteristics, whether password substitution should be used, restrictions on SSCP access, the system name associated with the LU, and security.

Local LU Configuration Parameters

The following parameters are required for local LU configuration:

LU name
The LU name of the local LU.

If you do not know what name to use, consult your SNA network planner.

This LU name is the second part of the fully qualified LU name of the local LU. The first part of the fully qualified LU name (the network name) is always the same as the first part of the CP name of the local node.

LU alias
The LU alias of the LU. If you do not enter an alias, the LU name is used as the alias.

Host LS/DLUR PU
The name of the host link station or DLUR PU to which the LU belongs. (This field applies only if the LU is a dependent LU.)

LU number
The LU number of the dependent LU. (This field applies only if the LU is a dependent LU.)

Member of default pool
Whether to make the LU a member of the default dependent APPC LU pool. An application that does not specify a particular local LU to use is assigned an available LU from the default pool.

This field applies only if the LU is a dependent LU.

Additional Configuration

After performing the local LU configuration, continue with the following configuration tasks:

- To define a remote node, see "Defining Remote Nodes."
- To define a partner LU, see "Defining Partner LUs" on page 83.
- To define an invokable TP, see "Defining TPs" on page 86.
- To define a mode, see "Defining Modes and Classes of Service" on page 92.
- To define CPI-C side information, see "Defining CPI-C Side Information" on page 96.
- To define APPC security, see "Configuring APPC Security" on page 98.
- To configure 5250 communication, see Chapter 7, "Configuring User Applications," on page 101.

Defining Remote Nodes

You must define a remote node (and the partner LUs on the node) in the following situations:

- If the local node is a LEN node, you must define all of the remote nodes and any partner LUs on the remote node with which it communicates using APPC. A LEN node is not able to dynamically locate partner LUs; the remote node definition enables it to do so.
- If the remote node is a LEN node and the local node is a network node that acts as the LEN node's network node server, you must define the LEN node (and its partner LUs) as a remote node on the network node server. This definition enables nodes in the rest of the APPN network to locate LUs on the LEN node.
- If the remote node is in a different APPN network, you must define the remote node because it cannot be dynamically located.

If you need to define the remote node and did not do so when you were defined the link station, you must do so before you can use APPC communications over the link.

When you add a remote node definition, a partner LU with the same name as the remote node is automatically added; this is the control point LU for the remote node. If your application uses this partner LU, you do not need to add another partner LU, although you may want to add an LU alias for the partner LU. To add an alias, double click on the partner LU and enter the alias in the Partner LU Configuration dialog.

If both the local node and the remote node are end nodes or network nodes and are part of an APPN network, partner LUs are located dynamically when needed. In this case, do not define the remote node where the LUs are located, because defining the node can cause the protocols in APPN that dynamically locate LUs to malfunction.

To prevent this malfunction, Communications Server for Linux does not permit you to define a remote node with which it has CP-CP sessions active (or with which it has had CP-CP sessions in the past). Additionally, if you have previously defined a remote node and Communications Server for Linux establishes CP-CP sessions with it, the entry is temporarily converted into a dynamic one. You should correct the fault by deleting the remote node definition when the node is inactive.

To configure a remote node, use one of the following methods:

Motif administration program
Select **APPC** and **New remote node** from the **Services** menu on the Node window.

Command-line administration program
To define a remote node, issue the following command:

define_directory_entry

To define a partner LU, issue the following command:

define_partner_lu

Remote Node Configuration Parameters

The following parameter is required for remote node configuration:

Node's SNA network name
The fully qualified CP name of the remote node. The value entered on this dialog must match the CP name configured at that remote node.

Additional Configuration

After performing the remote node configuration, continue with the following configuration tasks:

- To define a partner LU, see "Defining Partner LUs" on page 83.
- To define an invokable TP, see "Defining TPs" on page 86.
- To define a mode, see "Defining Modes and Classes of Service" on page 92.
- To define CPI-C side information, see "Defining CPI-C Side Information" on page 96.
- To define APPC security, see "Configuring APPC Security" on page 98.
- To configure 5250 communication, see Chapter 7, "Configuring User Applications," on page 101.

Defining Partner LUs

If both the local node and the remote node are network nodes, or if one is a network node and the other is an end node, and your application uses an LU name to refer to the partner LU, there is no need to define the partner LU, because it can be dynamically located using APPN. However, if your application uses an LU alias to refer to its partner LU, you should add a partner LU alias definition.

If either the local node or the remote node is a LEN node, you must define the partner LU as a child of the remote node, because a LEN node cannot take part in dynamic location of LUs. If your application uses the control point LU of the remote node as its partner LU, the control point LU was defined automatically when you defined the remote node.

You can use wildcards to configure multiple partner LUs that are all located on the same remote node and whose names start with the same characters. Using wildcards means that you do not need to configure each partner LU individually.

To configure a partner LU, use one of the following methods:

Motif administration program
You can use the Motif administration program to add a partner LU alias, add a definition of a partner LU on a specific remote node, or define multiple partner LUs using wildcards. Select **APPC**, **New partner LUs**, and one of the following from the **Services** menu on the Node window.

- **Partner LU alias**
- **Partner LU on remote node**
- **Wildcard partner LU on remote node**

Command-line administration program
To define a partner LU, issue the following command:

define_partner_lu

To define a LEN node as a partner LU, issue the following commands:

define_adjacent_len_node

define_directory_entry

Partner LU Configuration Parameters

The following parameters are required for partner LU configuration:

Partner LU name
The fully qualified LU name of the partner LU. This name must match the name that is configured at the remote node for this LU. If you do not know what that name is, consult your SNA network planner.

This field applies when you define partner LU on a specific remote node or when you define a partner LU alias.

Wildcard partner LU name
A name that matches the fully qualified LU names of multiple partner LUs. (This field applies only if you define partner LUs using wildcards.) The wildcard partner LU name consists of two strings, each of 1–8 characters:

- The first string can be a complete SNA network name that matches the first part of the fully qualified partner LU names exactly, or a wildcard

prefix that matches the beginning of the network name for the partner LUs. If you supply a wildcard prefix as the value for the first string, leave the second string blank. For example, a wildcard entry of **A** would match all LUs in the SNA networks named A, ANT, or APPN (but not BUFFALO or ZEBRA).

- If you supply a complete SNA network name for the first string, you can also enter a value for the second string. (You cannot specify the second string without supplying a valid SNA network name for the first string.) The second string is treated as a wildcard prefix, which must match the start of the second part of the fully qualified partner LU names. For example, a wildcard entry of **A.F** would match partner LUs names A.FRED or A.FREDDY (but not APPN.FRED or A.B).

If you leave both strings blank, the wildcard partner LU definition matches any partner LU name.

Alias A locally displayable alias for the partner LU. You do not have to specify an LU alias if there is no local application that refers to the partner LU using an LU alias.

This field applies when you define partner LU on a specific remote node or when you define a partner LU alias.

Uninterpreted Name
The uninterpreted name used by dependent local LUs when requesting the host to start an LU-LU session between the partner LU and the local LU. This name enables the partner LU name configured locally (and used by applications) to differ from the partner LU name configured on the host.

The default uninterpreted name is the second part of the partner LU name. This is correct in most cases. If in doubt, consult your SNA network planner.

This field applies when you define partner LU on a specific remote node or when you define a partner LU alias.

Supports parallel sessions
Whether the partner LU can support more than one session at a time. In most cases, the partner LU supports many sessions at one time, but some LEN nodes do not support parallel sessions.

This field applies when you define partner LU on a specific remote node or when you define a partner LU alias.

Location
The fully qualified CP name of the node on which the partner LU resides, or of a node that can provide access to the partner LU. If you supply the name of a remote node that has not yet been defined, you need to define it if you cannot discover the node dynamically.

This field applies only if you define a partner LU on a specific remote node.

Defining Link Station Routing for a Partner LU

You can use link station routing to identify the location of a partner LU by the link station that is used to reach it.

Note:

1. Link station routing is not required in an APPN network, where resources can be located dynamically. You are not recommended to use link station routing in an APPN network because it bypasses the normal APPN routing mechanisms.
2. You cannot use link station routing with an Enterprise Extender (HPR/IP) link station. This is because all traffic on this link type must flow over an RTP connection, which is not fixed to a particular link station and can switch to a different path.

To configure link station routing for a partner LU, use one of the following methods:

Motif administration program
Select **APPC**, **New partner LUs**, and **Partner LU on link station** from the **Services** menu on the Node window.

Command-line administration program
Issue the following command:

define_ls_routing

Link Station Routing Parameters

The following parameters are required for link station routing configuration:

LU name
The name of the local LU that controls the link station (if the partner LU is to be located through a specific link station).

LS name
The name of the link station.

Partner LU name
Either the fully qualified LU name of the partner LU or a wildcard name:

- A fully qualified LU name consists of two strings, each of 1–8 characters.

  This name must match the name that is configured at the remote node for this LU. If you do not know what that name is, consult your SNA network planner.
- A wildcard partner LU name matches the fully qualified LU names of multiple partner LUs. The wildcard partner LU name consists of two strings, each of 1–8 characters:
  - The first string can be a complete SNA network name that matches the first part of the fully qualified partner LU names exactly, or a wildcard prefix that matches the beginning of the network name for the partner LUs. If you supply a wildcard prefix as the value for the first string, leave the second string blank. For example, a wildcard entry of **A** would match all LUs in the SNA networks named A, ANT, or APPN (but not BUFFALO or ZEBRA).
  - If you supply a complete SNA network name for the first string, you can also enter a value for the second string. (You cannot specify the second string without supplying a valid SNA network name for the first string.) The second string is treated as a wildcard prefix, which must match the start of the second part of the fully qualified partner LU names. For example, a wildcard entry of **A.F** would match partner LUs names A.FRED or A.FREDDY (but not APPN.FRED or A.B).

  If you leave both strings blank, the wildcard partner LU definition matches any partner LU name.

Use partner LU name as a wildcard
Whether to use the partner LU name as a wildcard, rather than as a literal fully qualified LU name.

Additional Configuration

After performing the partner LU configuration, continue with the following configuration tasks:

- To define an invokable TP, see "Defining TPs."
- To define a mode, see "Defining Modes and Classes of Service" on page 92.
- To define CPI-C side information, see "Defining CPI-C Side Information" on page 96.
- To define APPC security, see "Configuring APPC Security" on page 98.
- To configure 5250 communication, see Chapter 7, "Configuring User Applications," on page 101.

Defining TPs

This section explains how to define an APPC TP.

In most cases, you do not need to define TPs that run on the Communications Server for Linux system; but you do need to configure a TP definition in the following cases:

APPC Characteristics
If the TP on the Communications Server for Linux computer is the invoking TP (or source TP—the TP that starts the APPC conversation) and you do not need to restrict access to the TP, you do not need to define the TP. You can, however, define an APPC TP, as described in "TP Definition Parameters" on page 91, to specify the following characteristics:

- To define conversation security for the TP.
- To indicate whether the TP uses basic or mapped conversations.
- To specify sync point processing.
- To specify handling of PIP data.

Invokable TPs
To enable a TP to be started automatically in response to an incoming allocation request, define it as an invokable TP as described in "TP Invocation Parameters on a Server" on page 88.

An invokable TP (or target TP) is one that is started in response to an incoming allocation request. You must create a TP definition for an invokable TP. An invokable TP can be an APPC TP that issues RECEIVE_ALLOCATE, or a CPI-C application that issues Accept_Conversation or Accept_Incoming.

Note: In this manual, the phrase "Receive_Allocate" is used to indicate any of these three API calls.

You can also define an invokable TP to route incoming allocation requests to a running TP.

For an invokable TP, you can also specify a timeout value, to limit the wait for an allocation request. (You can only configure this option using command-line administration.)

Communications Server for Linux uses the invokable TP definition for the following purposes:

- When a TP issues Receive_Allocate, Communications Server for Linux searches for an invokable TP definition with the appropriate TP name. If the definition exists, and includes a value for the Receive_Allocate timeout, Communications Server for Linux uses this value when processing the Receive_Allocate; otherwise it uses the default (no timeout, which causes the TP to wait indefinitely).
- When an incoming Allocate request arrives at the target system, and the requested TP is not already running with a Receive_Allocate outstanding, Communications Server for Linux searches for a TP definition with the TP name specified on the incoming Allocate. If the definition exists, Communications Server for Linux uses the information in this definition to start the TP (if multiple instances are permitted or the TP is not already running), or to determine that it should queue the incoming Allocate (if the TP is already running and multiple instances are not permitted).

If necessary, you can configure both types of definitions for the same TP (for example, to define conversation security for an invokable TP).

To configure a TP definition, use one of the following methods:

To define APPC characteristics:

Use either of the following methods:

Motif administration program

Select **APPC** and **Transaction Programs** from the **Services** menu on the Node window. When Communications Server for Linux displays the TP window, select the bottom pane and click on the **New** button, or select an existing TP definition and click on the **Properties** button.

Command-line administration program

Issue the **snaadmin define_tp** command.

To define an invokable TP:

The configuration methods for servers and clients are different:

- On a server, use either of the following methods:

  Motif administration program

  Select **APPC** and **Transaction Programs** from the **Services** menu on the Node window. When Communications Server for Linux displays the TP window, select the top pane and click on the **New** button, or select an existing invokable TP definition and click on the **Properties** button.

  Command-line administration

  Issue the **snatpinstall** command.

- UNIX

  On an IBM Remote API Client on AIX or Linux, issue the **snatpinstall** command.

- WINDOWS

On a Windows client, issue the **tpinst32** command. (This command applies to both 32–bit and x64 versions of Windows.)

For information about using the **snatpinstall** or **tpinst32** command, see Appendix B, “Configuring an Invokable TP from the Command Line,” on page 171.

TP Invocation Parameters on a Server

This section describes the parameters required by the Motif administration program or the command-line administration program when configuring an invokable TP on a server. For information about configuring an invokable TP on a client, see Appendix B, “Configuring an Invokable TP from the Command Line,” on page 171.

The following parameters are required for a TP that can be invoked on the local node:

TP name

A TP name in one of the following forms:

`Application TP`

If the TP is a user application, supply the name as normal characters (up to 64 characters in length).

`Service TP`

If the TP is an SNA service transaction program, enter the name in hexadecimal (up to eight hexadecimal digits, representing 4 bytes).

You can define multiple APPC invokable TPs that have the same TP name, provided each TP definition specifies a different LU alias. You cannot do this for CPI-C invokable TPs, because you cannot specify a particular LU alias to use; each CPI-C invokable TP must have a different name.

Parameters are for invocation on any LU/on specific LU

If the TP is an APPC TP, this parameter specifies whether to make the TP invokable on any LU or only on a specific LU. By default, the TP can be invoked on any LU.

Note: If the TP is a CPI-C application, this field must be set to make the TP invokable on any LU. CPI-C does not support accepting incoming Attaches from a particular local LU; attempting to specify this option for a CPI-C application will cause errors in routing the incoming Attach to the TP.

LU alias

This field must not be used if the TP is a CPI-C application. If the TP is an APPC application, it applies only if you specify that the parameters for this TP definition are for invocation on any LU.

The local LU alias from which the TP is to accept incoming Attaches. This name must match the name of a local APPC LU on the Communications Server for Linux node. If you do not specify an LU alias, the TP accepts incoming Attaches from any local LU.

If a non-blank LU alias is specified, the TP must use the extended form of the RECEIVE_ALLOCATE verb and specify this LU alias as a parameter to the verb. This enables Communications Server for Linux to route the

incoming Attach to the correct TP. For more information about the different forms of RECEIVE_ALLOCATE, refer to *Communications Server for Linux APPC Programmer's Guide*. If you need to permit the TP to determine the correct LU alias at run-time rather than building it into the application, you can do this by setting an environment variable to contain the appropriate LU alias (using the *Environment* parameter), and designing the application to read this environment variable in order to determine how to issue RECEIVE_ALLOCATE.

You can define multiple TPs that have the same TP name, provided each TP definition specifies a different LU alias.

Multiple instances supported

If you do not select this option, the TP is a queued TP. Any incoming Allocate requests arriving while the TP is running are queued until the TP issues another Receive_Allocate, or until it finishes running and can be restarted. An incoming Allocate request is routed to this TP only if it is received by an LU that is configured to route incoming Allocate requests to this computer, or if it is received by an LU on this computer that has no routing information configured.

If you select this option, the TP is a nonqueued TP. Communications Server for Linux starts a new copy of the TP each time an incoming Allocate request arrives for it. A nonqueued TP cannot be started by an operator; it is always started automatically by Communications Server for Linux. For a nonqueued TP, Communications Server for Linux permits more than one copy of the TP to be running at a time. All copies run with the same user and group IDs and the same working directory, as defined by the *User ID* and *Group ID* parameters. If the TP writes to files on the local system, you need to ensure that different copies of the TP do not overwrite each other's files.

After a nonqueued TP has ended a conversation, it may terminate, or it may issue another RECEIVE_ALLOCATE. For frequently-used programs, this provides a way of avoiding the performance overhead of starting a new instance of the program for each conversation. Each time an Attach is received for a nonqueued, automatically started TP, Communications Server for Linux checks whether there is already a RECEIVE_ALLOCATE outstanding from an instance of this TP. If so, this TP is used for the incoming conversation; otherwise, Communications Server for Linux starts a new instance of the program.

Route incoming Allocates to running TP

This option applies only if multiple instances are not supported.

Select this option if the TP is a broadcast queued TP. Any incoming Allocate requests arriving while the TP is running are queued until the TP issues another Receive_Allocate, or until it finishes running and can be restarted. When the TP is started, information about the TP is broadcast to all servers on the LAN; if an LU on another computer receives an incoming Allocate request and has no routing information configured, it can dynamically locate the TP and route the Allocate request to it.

Using this option avoids having to configure explicit routing information on LUs, and enables load-balancing by running more than one copy of the same TP on different computers. However, if you want to avoid broadcasting information in order to reduce LAN traffic, or if you need to ensure that incoming Allocate requests arriving at a particular LU are always routed to the same copy of the TP, do not select this option.

Full path to TP executable
The full path and file name of the executable file for this TP.

The file must have execute permission for the user specified by the *User ID* parameter. In addition, if the executable file is to be run with *User ID* set to root, the file must be owned by root and must have `setuid` and `setgid` permission in order to be started automatically by Communications Server for Linux.

Arguments
Any command-line arguments to be passed to the TP, separated by spaces. The arguments are passed to the TP in the same order as they appear here.

This value is optional. If it is not included, the TP is invoked without any command-line arguments.

User ID
The user ID that Communications Server for Linux uses to start the TP. This line is required, and must be specified. The ID must be a valid Linux login ID on the Communications Server for Linux computer.

The TP is started in the home directory associated with this user ID. This home directory is also the default path for trace files and any other files accessed by the TP (unless the application overrides it by specifying a full path). If the application specifies a file name without a path, Communications Server for Linux searches for the file in this home directory; if the application specifies a file name with a relative path, Communications Server for Linux searches for the file in the specified directory relative to this home directory.

The executable file for the TP, specified by the *Full path to TP executable* parameter, must have execute permission for the specified user. In addition, if *User ID* is set to root, the file must be owned by root and must have `setuid` and `setgid` permission in order to be started automatically by Communications Server for Linux.

Group ID
The group ID that Communications Server for Linux uses to start the TP. This must be a valid Linux group ID on the Communications Server for Linux computer.

This parameter is optional. If it is not included, the default is `sna`.

Standard input
Specify the full path name of the standard input file or device used by the TP.

This parameter is optional. If it is not included, the default is `/dev/null`.

Standard output
Specify the full path name of the standard output file or device used by the TP.

This parameter is optional. If it is not included, the default is `/dev/null`.

Standard error
Specify the full path name of the standard error file or device used by the TP.

This parameter is optional. If it is not included, the default is `/dev/null`.

Environment
Specify any environment variables required by the TP.

Each variable is specified in the form *environment_variable=value*, and can be up to 255 characters long. The string *environment_variable=value* must not contain space or tab characters before or after the = character.

In the Motif administration program, if you need to specify more than one environment variable (up to a maximum of 64), separate them using the | character. The variables are set in the same order as they appear here.

If the TP is a CPI-C application, note that you cannot set the environment variable `APPCLLU` using this field. The local LU cannot be specified for an automatically-loaded CPI-C application.

This field is optional. If it is not included, no environment variables are used.

TP Definition Parameters

You can configure an APPC TP to specify conversation security, conversation type, sync level, and handling of PIP data. The following parameters are required to define a TP for APPC communication:

TP name
A TP name in one of the following forms:

`Application TP`
If the TP is a user application, supply the name as normal characters (up to 64 characters in length).

`Service TP`
If the TP is an SNA service transaction program, supply the name in hexadecimal (up to eight hexadecimal digits, representing 4 bytes).

Conversation level security required
Select this option if an allocation request must include a valid user name and password (or an indicator that the password has already been verified). If you do not select this option, no verification is required.

Restrict access
Select this option if the user name must be included on a security access list. This field applies only if the *Conversation level security required* option is selected.

Security access list
Name of a security access list that contains user IDs permitted to access this TP. If the *Restrict access* option is selected, you must provide this value.

Conversation type
Specify whether the TP accepts only basic conversations, only mapped conversations, or either type of conversation.

Sync level
Specify the levels of confirm synchronization that the TP accepts. For more information on confirm synchronization, refer to the Communications Server for Linux APPC Programmer's Guide. Select one of the following values:

- `None`
- `Confirm`
- `Sync-point`
- `None or Confirm`
- `None, Confirm or Sync-point`

PIP allowed
Select this option if the TP accepts PIP data (Program Initialization Parameters).

Defining Modes and Classes of Service

A mode specifies a set of characteristics that a local LU (LU type 6.2) uses to communicate with its partner LU. These characteristics include information about the way data is transmitted between the two LUs (such as maximum RU lengths and pacing window sizes), and about whether the LUs can establish parallel sessions.

In addition, you may need to specify requirements for the communication path between the LUs, such as enforcing a certain level of network security, minimizing transmission time, or avoiding the use of expensive communication links. You can define these requirements using a class of service (COS), which specifies minimum and maximum acceptable values for characteristics such as transmission time, transmission cost, and network security. The COS also specifies weightings associated with different ranges of these values. This enables the node to calculate the best route across the network when two or more routes to the same remote LU are available.

If the Communications Server for Linux node is a network node, the definition of each mode includes the name of the required COS for that mode. If the Communications Server for Linux node is a LEN node or end node, you do not need to associate a COS with the mode; the COS name is determined dynamically.

SNA defines a number of standard modes and associated COSs that cover the requirements of most systems; you generally do not need to define additional modes and COSs. You need to define a mode only if the required mode is not one of the predefined standard modes, which can be viewed in the Modes window.

The default mode is used if the mode name in an incoming conversation is unrecognized. If you do not specify a default mode, the default mode is the blank mode name.

The standard mode names and their associated COS names are shown in Table 2. For more information about the parameters associated with these standard names, refer to the IBM SNA manuals *LU 6.2 Reference—Peer Protocols* (for modes) and *APPN Architecture Reference* (for COSs).

Table 2. Standard Mode and COS Names

Mode Name	Associated COS Name	Purpose
(blank)	#CONNECT	Sessions that do not specify a mode name (basic default COS parameters)
#BATCH	#BATCH	Sessions used by batch-processing applications
#BATCHSC	#BATCHSC	Sessions used by batch-processing applications, with a minimal level of routing security
#BATCHC	#BATCH	Sessions using compression in batch-processing applications
#BATCHCS	#BATCH	Sessions using compression in batch-processing applications, with a minimal level of routing security
#INTER	#INTER	Sessions used by interactive applications

Table 2. Standard Mode and COS Names (continued)

Mode Name	Associated COS Name	Purpose
#INTERSC	#INTERSC	Sessions used by interactive applications, with a minimal level of routing security
#INTERC	#INTER	Sessions using compression in interactive applications
#INTERCS	#INTER	Sessions using compression in interactive applications, with a minimal level of routing security
SNASVCMG	SNASVCMG	CNOS (change number of sessions) and management services sessions
CPSVCMG	CPSVCMG	CP-CP sessions between nodes
CPSVRMGR	CPSVRMGR	CP-CP sessions used for dependent LU requester (DLUR)
QPCSUPP	#CONNECT	Sessions used for 5250 emulation

Once a mode has been configured, it can be used by any APPC or CPI-C application to activate a session between a local LU and a partner LU. An APPC application must specify the mode to use, but a CPI-C application can use CPI-C side information (which includes the mode name). For more information about configuring CPI-C side information, see "Defining CPI-C Side Information" on page 96.

To configure a mode or class of service, use one of the following methods:

Motif administration program
Select **APPC** and **Modes** from the **Services** menu on the Node window, then choose **New** on the Mode window.

Command-line administration program
To define a mode, issue the following command:

define_mode

To change the default mode, issue the following command:

define_defaults

To define a class of service, issue the following command:

define_cos

Mode Configuration Parameters

The following parameters are required for mode configuration:

Name The name of the mode you are defining. The mode name is a string of 1–8 characters.

APPC applications that use this mode, including both local and remote applications, may also use this name, so check the name with your application developer (or refer to your product documentation for a third-party application).

COS name
The name of the class of service for this mode. The name is a string of 1–8

characters. Usually you can simply specify #INTER for modes used for interactive data exchange and #BATCH for modes used for bulk data transfer.

This field applies only to a network node.

If you do not know what value to specify, consult your SNA network planner.

Session limits

Use the following fields to specify session limits:

Initial session limit

The maximum number of sessions (up to the maximum session limit) that a pair of LUs can have using this mode, unless a different maximum is negotiated using CNOS.

Normally, use the value 8 for this field. If you are in doubt, consult your SNA network planner or APPC application developer (or for a third-party application, the product documentation).

Maximum session limit

The maximum number of sessions (up to 32,767) permitted between a pair of LUs using this mode, even with CNOS negotiation.

This field is usually set to the same value as the initial session limit. If you are in doubt, consult your SNA network planner or APPC application developer (or for a third-party application, the product documentation).

Minimum contention winner sessions

The number of sessions (up to the session limit) that Communications Server for Linux must reserve for use by the local LU as the contention winner.

This field can usually safely be set to 0, but if you are not sure, consult your SNA network planner.

The sum of the minimum contention winner sessions and the minimum contention loser sessions must not exceed the initial session limit.

Minimum contention loser sessions

The minimum number of sessions that Communications Server for Linux must reserve for use by the local LU as the contention loser. Together with the value in the *Minimum contention winner sessions* field, this value determines how to resolve contention for a session.

This can usually safely be set to 0, but if you are not sure, consult your SNA network planner.

The sum of the minimum contention winner sessions and the minimum contention loser sessions must not exceed the initial session limit.

Auto-activated sessions

The number of sessions (up to the minimum contention winner count) that are automatically activated after CNOS negotiation has taken place for a session between a local LU and partner LU using this mode. Specifying a value for this field enables an LU that uses this mode to start sessions automatically in response to a request from a TP for a conversation to be allocated immediately.

Receive pacing window
Use these fields to specify how many RUs can be received before an SNA pacing response is sent:

Initial window size
The initial setting for the number of request units (RUs) that the local LU can receive before it must send a pacing response to the remote LU. This can be safely set to `4`.

Setting it higher can improve performance in some circumstances, but doing so also increases memory usage.

Maximum window size
The maximum number of request units (RUs) that the local LU can receive before it must send a pacing response to the remote LU.

This value is optional. If it is not supplied, the maximum receive pacing window is unlimited. If a value is supplied, it is used to limit the size of the receive pacing window for adaptive pacing. If adaptive pacing is not used, this value is ignored.

The pacing window can be from `0` through `32767` bytes. A value of `0` specifies an unlimited window.

If the adjacent node supports only fixed pacing, these values determine the fixed-pacing window size; but the adjacent node can still set a window size through negotiation. If the adjacent node uses adaptive pacing, these values set the initial window size.

Specify timeout
Select this option if you want to specify the number of seconds (`0 - 65535`) that an LU 6.2 session using this mode must be inactive before it can time out. Changing this value affects only sessions that are activated using this definition (not sessions that are already active).

If you use a value of `0`, sessions are timed out as soon as they become free.

Restrict maximum RU size
Select this option if you want to specify the maximum RU size, which determines how much data is buffered before being sent to the partner LU.

The upper limit can be from `256` through `62440` bytes. You can safely set the upper limit to `1024` bytes. Setting it higher can improve performance in some circumstances, but doing so also increases memory usage.

The lower limit can be 0 or a value from 256 through the upper limit you specify.

If the value in this field is different from the RU size defined for the remote node, the size used for a session with that node can be negotiated to establish an appropriate RU size for the session. The actual value cannot be lower than the lower limit field.

These numbers, together with the send and receive pacing values, can be used to tune the session-level throughput between the local and partner LUs. If you do not know what values to use, start with the default values and adjust them as needed to maximize throughput.

Compression supported
Whether data compression is supported for sessions using this mode. If you do not set this option, compression will not be used.

If you set this option, you can specify the maximum compression levels to be used for inbound data and for outbound data. These are separate options so that you can specify different levels for the two directions, or use compression in one direction but not in the other. In each direction, you can select `None` for no compression, or one of the values `RLE` (minimum compression), `LZ9`, or `LZ10` (maximum compression).

Reset to SNA defined values
If you are modifying a standard mode using the Motif dialog, you can click on this button to reset the values of the mode parameters to the SNA-defined values.

Additional Configuration

After performing the mode configuration, continue with the following configuration tasks:

- To define CPI-C side information, see "Defining CPI-C Side Information."
- To define APPC security, see "Configuring APPC Security" on page 98.
- To configure 5250 communication, see Chapter 7, "Configuring User Applications," on page 101.

Defining CPI-C Side Information

If you are supporting a CPI-C application that uses CPI-C symbolic destination names, you need to define the CPI-C side information. The side information associates the symbolic destination name with information about the partner TP, partner LU, mode, and security for the conversation.

To determine the symbolic destination name for CPI-C, consult the application developer (or for a third-party application, consult the product documentation).

To configure CPI-C side information, use one of the following methods:

Motif administration program
Select **APPC** and **CPI-C** from the **Services** menu on the Node window.

Command-line administration program
Issue the following command:

define_cpic_side_info

CPI-C Configuration Parameters

For each CPI-C symbolic destination name used by the application, collect the following information:

Name The symbolic destination name used by the CPI-C applications (also known as TPs) that you want to run. This name can be 1–8 characters in length.

The application developer (or for a third-party application, the product documentation) can provide this name.

Local LU
The local LU for any conversations initiated by TPs using this side information using one of the following methods:

Local LU alias
An alias for a local LU.

Use default LU
Specify this option to use a member of the default pool (if one exists) or the node control point LU (if no default pool is defined).

If the APPCLLU environment variable is set, the local LU information you supply is ignored, and the LU specified for the environment variable is used instead.

Partner LU
Either an alias or the fully qualified partner LU name for conversations initiated by local TPs using this side information. The partner LU must be an LU that is configured on the computer that runs the partner TP.

Mode The name of the APPC mode that is to be used to access the partner LU. In most cases, the mode is one of the following predefined modes:

- A blank name
- #BATCH
- #BATCHSC
- #INTER
- #INTERSC
- QPCSUPP

Partner TP
The name of the transaction program with which the CPI-C application communicates:

- If the TP is a user application, specify the name as normal characters (up to 64 characters in length).
- If the TP is a service TP, specify the name in hexadecimal (up to 8 hexadecimal digits, representing 4 bytes).

The application developer (or for a third-party application, the product documentation) can provide this information.

Security
The level of conversation-level security you want to use. The options are as follows:

None The partner TP does not require security parameters to be checked.

Same The partner TP uses security, but accepts verification by the local TP of the user ID and password provided by the initiating TP. If you choose a security level of Same, you also need to specify a valid user ID that is accepted by the partner TP.

Program
The partner TP requires a User ID and password. If you choose a security level of Program, you need to specify a valid user ID and password that are accepted by the partner TP.

Program strong
The partner TP requires a user ID and password. Both the local and remote nodes must support security enhancements so that the password is encrypted.

Refer to the documentation for the CPI-C application or consult the application programmer to find out what security parameters to use.

User ID
If you have chosen a security level of Same, Program, or Program strong,

specify a user ID to be sent on the initiating message to the remote application. This value must match a user ID that the application is defined to accept.

This user ID is not related to Linux login user IDs on either the local or the remote node. If the remote node is running Communications Server for Linux, the user ID must be configured on the remote node using the Conversation Security Configuration dialog.

Password
If the security level is specified as `Program` or `Program strong`, specify a password to be sent when the conversation is allocated. This value must match the password defined at the remote application for use with the supplied user name.

This password is not related to Linux login passwords on either the local or the remote node. If the remote node is running Communications Server for Linux, the password must be configured on the remote node using the Conversation Security Configuration dialog.

Additional Configuration

After performing the CPI-C configuration, continue with the following configuration tasks:

- To define APPC security, see "Configuring APPC Security."
- To configure 5250 communication, see Chapter 7, "Configuring User Applications," on page 101.

Configuring APPC Security

You can perform the following configuration tasks for APPC security:

- Configuring session security as described in "Configuring Session Security"
- Configuring conversation security as described in "Configuring Conversation Security" on page 99
- Configuring security access lists as described in "Configuring a Security Access List" on page 99

Configuring Session Security

Session-level security is used to validate LU-LU sessions. Each definition consists of a local LU name, a partner LU name, and a password.

Communications Server for Linux uses the password to validate sessions between the local LU and partner LU. (The passwords are not related to Linux logon passwords.)

To configure session security, use one of the following methods:

Motif administration program
Select **APPC**, **Security**, and **Session-level security** from the **Services** menu on the Node window.

Command-line administration program
Issue the following command:

define_lu_lu_password

Session Security Configuration Parameters

The following parameters are required for session security configuration:

Local LU
: The LU name of the local LU. The name is a string of 1–8 characters.

Partner LU
: The fully qualified LU name of the partner LU.

Password
: A password that Communications Server for Linux can use to validate sessions between the local LU and the partner LU. The password is a 16-digit hexadecimal number that is used to create a key, which is exchanged when the session is established. This password is not related to Linux login passwords on either the local or the remote node.

Additional Configuration

After performing the session security configuration, continue with the following configuration tasks:

- To configure conversation security, see "Configuring Conversation Security."
- To configure 5250 communication, see Chapter 7, "Configuring User Applications," on page 101.

Configuring Conversation Security

Conversation security is used to validate incoming conversations. Each definition consists of a user ID and a password. This user ID is not related to Linux login user IDs on either the local or the remote node.

To configure conversation security, use one of the following methods:

Motif administration program
: Select **APPC**, **Security**, and **Conversation-level security** from the **Services** menu on the Node window.

Command-line administration program
: Issue the following command:

  define_userid_password

Conversation Security Configuration Parameters

The following parameters are required for conversation security configuration:

User ID
: The user ID to be accepted in an incoming conversation from a remote node. The user ID can be up to 10 characters long.

Password
: The password to be accepted in an incoming conversation from a remote node. The password can be up to 10 characters long.

Additional Configuration

After configuring conversation security, you can configure 5250 communication as described in Chapter 7, "Configuring User Applications," on page 101.

Configuring a Security Access List

You can define an APPC security access list to control access to an LU or TP (or both). This list can be referred to by the definition for an APPC local LU or TP.

To configure a security access list, use one of the following methods:

Motif administration program
Select **APPC**, **Security**, and **Conversation-level security** from the **Services** menu on the Node window, then select the Security Access Lists pane and choose **New**.

Command-line administration program
Issue the following command:

define_security_access_list

Security Access List Configuration Parameters

The following parameters are required for security access list configuration:

Name Name of the security access list. The definition for an APPC TP or local LU can use this name to refer to the access list.

Users in access list
The names of users included in the security access list.

Additional Configuration

After performing the security access list configuration, continue with the following configuration tasks:

- Configure TP access as described in "Defining TPs" on page 86.

Chapter 7. Configuring User Applications

This chapter provides instructions for configuring SNA resources to support user applications that use any of the following communication: 3270, 5250, and LUA. The SNA resources required by such applications are LUs.

For 3270, LUA, and dependent APPC communication, you must configure dependent LUs. For independent APPC and 5250 communication, you can use the default control point LU (defined automatically when you configure the local node) or define independent LUs.

Before you can configure the resources described in this chapter, you must perform the following configuration:

- Configure the node as described in "Configuring the Node" on page 54.
- Configure connectivity as described in Chapter 4, "Defining Connectivity Components," on page 59. For 3270, LUA, and dependent APPC communication, you must configure the link to support dependent LU traffic. For independent APPC and 5250 communication, the link must support independent LU traffic.

  You do not need to configure a direct link to the host if you are using upstream SNA gateway or DLUR. For more information, see "Configuring SNA Gateway" on page 111 and "Defining DLUR PUs" on page 71.

The following list describes the configuration tasks required for each type of user application:

3270 applications

For 3270 communication, configure the following resources:

1. For a 3270 display or printer, define a dependent LU as described in "Defining LU Types 0–3" on page 75.
2. To enable 3270 displays to select from a pool of LUs, define an LU pool as described in "Defining LU Pools" on page 77. If a display uses a dedicated LU, you can skip this step.

5250 applications

For 5250 communication, configure the following resources:

1. Configure the node for APPC communication:
   a. If you can use the local node's control point LU, you do not need to configure a local LU. If you need a local LU definition (for example, to use session security), define the local LU as described in "Defining Local LUs" on page 80.
   b. If the local node is a LEN node, you must define the AS/400 system as a remote node as described in "Defining Remote Nodes" on page 81.

      If the local node is an APPN end node or network node, you can use the control point LU on the AS/400 system as a partner LU, so you do not need to configure any other partner LUs.

   You do not need to define any modes, because 5250 uses the standard mode `QPCSUPP`.

LUA applications

To support an LUA application, configure the following resources:

1. Define a dependent LU as described in “Defining LU Types 0–3” on page 75.
2. To enable an LUA application to select from a pool of LUs, define an LU pool as described in “Defining LU Pools” on page 77. If the application uses a dedicated LU, you can skip this step.

An LUA application uses the LU 0–3 resources of the node to communicate with a host application. You do not need to define any additional resources.

Chapter 8. Configuring Passthrough Services

Passthrough services on a server running Communications Server for Linux enable communication between an SNA host and local systems that are not directly connected to the host.

Communications Server for Linux includes TN server support for TN3270, TN3287, and TN3270E clients, collectively referred to as "TN3270 clients." To configure this function, see "Configuring TN Server."

Communications Server for Linux also includes TN Redirector support for passthrough TCP/IP host access to TN3270, TN3270E, TN5250 and VT clients, referred to collectively as "Telnet clients". To configure this function, see "Configuring TN Redirector" on page 107.

SNA gateway provides connectivity between the host and local systems. You can configure LUs on the local node to support this function (see "Configuring SNA Gateway" on page 111) or you can define a template that is used to support downstream LUs that have not been explicitly configured (see "Defining DLCs, Ports, and Connection Networks" on page 60).

DLUR supports dependent LU sessions between the host and nodes in an APPN network. To configure this function, see "Configuring DLUR" on page 113.

Configuring TN Server

TN server enables TN3270 clients to communicate with a host through an intermediate Communications Server for Linux node that implements the TN server. The TN3270 clients connect to the TN server using TCP/IP, and use LUs defined on the TN server. The TN server LUs establish sessions with LUs at the host to support TN3270 sessions for the clients.

Before you can configure TN server, you must perform the following configuration tasks:

- Define the local node as described in "Configuring the Node" on page 54.
- Configure a port and link station for dependent traffic between the local node and the host, as described in Chapter 4, "Defining Connectivity Components," on page 59.
- Define the TN3270 LUs on the local node that are used for communication with the host. To add the LUs, see "Defining LU Types 0–3" on page 75.
- If you are going to use any LU pools, define them as described in "Defining LU Pools" on page 77.

To configure TN server, perform the following tasks:

- Configure a TN server access record for each TN3270 client who will use the server, or a default record that enables any client to access the server (see "Configuring TN Server Access Records" on page 104).
- If you are supporting TN3270E or TN3287 clients, you can define an association record for display and printer LUs (see "Configuring TN Server Association Records" on page 107). This record enables a TN3270E or TN3287 client to select

a specific printer (by selecting the associated display LU). The client must be authorized to select an LU in the TN server access record.

Additional options for TN server enable you to force printer responses, specify a keep-alive method for all TN3270 sessions, and specify how to access the external LDAP server that holds a revocation list used to check authorization for TN3270 clients. To access these options, use the **Services** menu on the TN Server window.

Configuring TN Server Access Records

TN server access records indicate which TN3270 clients can access the TN server and which LUs they should use. Each access record identifies a TN3270 client that is permitted to access the TN server, the TCP/IP port that the client connects to, and the LU or LU pool that the client uses.

You can also define a default record that enables access by any TN3270 client (with the same LUs or LU pools for all clients).

TN3270 clients can use the TN server only when the node, port, and link station are active.

To configure a TN server access record, use one of the following methods:

Motif administration program
Select **TN server** from the **Services** menu on the Node window, and **TN server** from the submenu. On the resulting window, select the TN Server Client Access Permissions pane and choose **New**.

Command-line administration program
Issue the following command:

define_tn3270_access

Note: If you define a TN server access record using the command-line administration program, **snaadmin**, or a NOF application, you can use the *listen_local_address* parameter to specify an address on the local TN Server computer to which the TN3270 client will connect. If you do this, the access record will not be displayed in the Motif administration program, so you cannot use that program to view or manage it. You can still manage it using the command-line administration program or a NOF application.

TN Server Access Record Configuration Parameters

The following parameters are required for TN server access record configuration:

TN3270 client address
The address that identifies the TN3270 client to which the access record applies:

Default record
Permit access by any TN3270 client.

TCP/IP name or alias
Permit access by a named TN3270 client. If you know the TCP/IP name of the client, select this option and enter the name. On many computers, you can find out the computer's TCP/IP name using the **hostname** command.

TCP/IP address
Permit access from a specific TCP/IP address. If you know the

TCP/IP address of the TN3270 client, select this option and enter the address. This can be either of the following.

- An IPv4 dotted-decimal address (such as `193.1.11.100`).
- An IPv6 colon-hexadecimal address (such as `2001:0db8:0000:0000:0000:0000:1428:57ab` or `2001:db8::1428:57ab`).

Support TN3270E
The level of TN3270 support provided by the node:

TN3270 Support only the TN3270 protocol. Selecting this option disables server support for TN3270E protocols, even if they are supported on the client.

TN3270E
Support both TN3270 and TN3270E protocols (the default).

TN3270 and TN3287 protocols are always supported, regardless of which option you choose.

For an AS/400 TN3270 client, this option must be set to `TN3270E`.

TCP/IP port number
The TCP/IP port number (on the TN server) for the port to which the TN3270 client connects.

Note: TCP/IP ports are completely unrelated to SNA ports.

The well-known port number for the TN3270 service is `23`. If you choose a different port number that is not in use on the TN server, you also need to configure that port number on the TN3270 clients (or start the TN3270 clients using an option to specify the port number). Port numbers above `2000` are likely to be available. Port numbers in the range `256–1023` may give slightly better security, but are more likely to be in use.

If you want a TN3270 client to be able to use more than one LU or LU pool, define multiple access records, each with a different TCP/IP port number, so that you can identify the different LUs or LU pools by specifying different port numbers.

Display LU assigned
The name of the LU that the TN3270 client accesses when it is active. The LU must be a dependent LU on the local node. You can specify the name of an LU pool rather than the name of a particular LU.

Printer LU assigned
The name of the default printer LU or LU pool for clients that use this access record. This LU must be defined as a dependent LU on the local node.

Allow access to specific LU
Specify this option to enable TN3270E and TN3287 clients to request a specific LU for a session. (This option is not available to TN3270 clients.)

SSL secure session
Specify this option to indicate that this session uses Secure Sockets Layer (SSL) to access the server.

This option is available only if you have installed the additional software required to support SSL on the server; otherwise you cannot select it.

Note: If this session's *TCP/IP port number* parameter indicates that it uses the Telnet daemon's TCP/IP port, do not use SSL for this session. If you use SSL on a session that uses the Telnet daemon's TCP/IP port, Telnet clients will not be able to use **telnet** to access the Communications Server for Linux computer while the node is active.

Perform client authentication

This option appears only if you have selected the *SSL secure session* option.

Specify this option to indicate that the TN Server requires the session to use client authentication. The client must send a valid certificate (information identifying it as a valid client authorized to use the TN Server).

As well as checking that the certificate is valid, the TN Server may also need to check the certificate against a certificate revocation list on an external LDAP server, to ensure that the user's authorization has not been revoked. In this case, you also need to use the TN Server Advanced Parameters dialog to specify how to access this server.

Security level

Indicates the SSL security level required for this session. The session will use the highest security level that both client and server can support; if the client cannot support the requested level of security or higher, the session will not be started.

This option appears only if you have selected the *SSL secure session* option.

Possible values are:

Authenticate Only
Certificates must be exchanged, but encryption will not be used. This option is typically used to avoid the overhead of encryption when the client is connecting across a secure intranet.

Authenticate Minimum
The client must request a certificate from the server to check its validity; encryption is not required (but can be used if the client requests it).

40 Bit Minimum
The client must support at least 40–bit encryption.

56 Bit Minimum
The client must support at least 56–bit encryption.

128 Bit Minimum
The client must support at least 128–bit encryption.

168 Bit Minimum
The client must support at least 168–bit encryption.

Note: Using encryption requires additional software to be installed with Communications Server for Linux; see *Communications Server for Linux Quick Beginnings* for more information. Depending on your location, you may not be able to use all the encryption levels listed because the software required to support them is not available in your country.

Additional Configuration

After performing the TN server access configuration, continue with the following configuration tasks:

- Configure TN server association records as described in "Configuring TN Server Association Records."

Configuring TN Server Association Records

A TN server association record defines an association between a printer LU and display LU, so that the TN3270E or TN3287 protocol can connect the two. If the access record for the client permits selection of a specific LU, this record enables a client to select a specific printer by specifying the associated display LU.

To configure a TN server association record, use one of the following methods:

Motif administration program
Select **TN Server** from the **Services** menu on the Node window, then select the Association Records pane on the TN Server window and choose **New**.

Command-line administration program
Issue the following command:

define_tn3270_association

TN Server Association Record Configuration Parameters

The following parameters are required for TN server association record configuration:

Display LU
The name of the display LU (which must be defined on the local node).

Printer LU
The name of the printer LU (which must be defined on the local node). Do not specify a printer LU that has been entered on another TN server association record.

Configuring TN Redirector

TN Redirector enables TN3270, TN3270E, TN5250 and VT clients, collectively known as Telnet clients, to communicate with a host through an intermediate Communications Server for Linux node that implements the TN redirector. The clients connect to the TN redirector using TCP/IP; the TN redirector then establishes a separate TCP/IP connection to the host.

To configure TN Redirector, perform the following tasks:

- Configure a TN Redirector access record for each Telnet client who will use the server, or a default record that enables any client to access the server (see "Configuring TN Redirector Access Records").

Configuring TN Redirector Access Records

TN redirector access records indicate which Telnet clients can access the TN redirector over a TCP/IP link. Each access record identifies a Telnet client that is permitted to access the TN redirector, the TCP/IP port that the client uses to connect to Communications Server for Linux, the TCP/IP port that Communications Server for Linux uses to connect to the host, and the SSL security settings. You can also define default records that enable access by any client.

If you want to permit any client to use the TN redirector and you want all clients to use the same host access configuration, you can define a default record.

Telnet clients can use the TN redirector only when the node is active.

To configure a TN redirector access record, use one of the following methods:

Motif administration program
Select **TN server** from the **Services** menu on the Node window, and **TN server** from the submenu. On the resulting window, select the TN Redirector Client Access Permissions pane and choose **New**.

Command-line administration program
Issue the following command:

define_tn_redirect

Note: If you define a TN redirector access record using the command-line administration program, **snaadmin**, or a NOF application, you can use the *listen_local_address* parameter to specify an address on the local TN Server computer to which the TN3270 client will connect. If you do this, the access record will not be displayed in the Motif administration program, so you cannot use that program to view or manage it. You can still manage it using the command-line administration program or a NOF application.

TN Redirector Access Record Configuration Parameters

TN redirector access record configuration consists of two groups of parameters, for the client and host TCP/IP connections.

The client parameters are as follows:

Telnet client address
The address that identifies the Telnet client to which the access record applies:

Default record
Permit access by any Telnet client.

TCP/IP name or alias
Permit access by a named Telnet client. If you know the TCP/IP name of the client, select this option and enter the name. On many computers, you can find out the computer's TCP/IP name using the **hostname** command.

TCP/IP address
Permit access from a specific TCP/IP address. If you know the TCP/IP address of the client, select this option and enter the address. This can be either of the following.

- An IPv4 dotted-decimal address (such as `193.1.11.100`).
- An IPv6 colon-hexadecimal address (such as `2001:0db8:0000:0000:0000:0000:1428:57ab` or `2001:db8::1428:57ab`).

TCP/IP port number
The TCP/IP port number (on the TN server) for the port to which the client connects.

Note: TCP/IP ports are completely unrelated to SNA ports.

You also need to configure this port number on the clients (or start the clients using an option to specify the port number). Port numbers above 2000 are likely to be available. Port numbers in the range 256–1023 may give slightly better security, but are more likely to be in use.

SSL secure session
Specify this option to indicate that this session uses Secure Sockets Layer (SSL) to access the server.

This option is available only if you have installed the additional software required to support SSL on the server; otherwise you cannot select it.

Perform client authentication
This option appears only if you have selected the *SSL secure session* option.

Specify this option to indicate that the TN Server requires the session to use client authentication. The client must send a valid certificate (information identifying it as a valid client authorized to use the TN Server).

As well as checking that the certificate is valid, the TN Redirector may also need to check the certificate against a certificate revocation list on an external LDAP server, to ensure that the user's authorization has not been revoked. In this case, you also need to use the TN Server Advanced Parameters dialog (accessed from the **Services** menu on the TN Server window) to specify how to access this server.

Security level
Indicates the SSL security level required for the client session. The session will use the highest security level that both client and server can support; if the client cannot support the requested level of security or higher, the session will not be started.

This option appears only if you have selected the *SSL secure session* option.

Possible values are:

Authenticate Only
Certificates must be exchanged, but encryption will not be used. This option is typically used to avoid the overhead of encryption when the client is connecting across a secure intranet.

Authenticate Minimum
The client must request a certificate from the server to check its validity; encryption is not required (but can be used if the client requests it).

40 Bit Minimum
The client must support at least 40–bit encryption.

56 Bit Minimum
The client must support at least 56–bit encryption.

128 Bit Minimum
The client must support at least 128–bit encryption.

168 Bit Minimum
The client must support at least 168–bit encryption.

Note: Using encryption requires additional software to be installed with Communications Server for Linux; see *Communications Server for Linux Quick Beginnings* for more information. Depending on your

location, you may not be able to use all the encryption levels listed because the software required to support them is not available in your country.

The destination host parameters are as follows:

Address
The address that identifies the host to which the access record applies:

TCP/IP name or alias
Access to a named host. If you know the TCP/IP name of the host, select this option and enter the name. On many computers, you can find out the computer's TCP/IP name using the **hostname** command.

TCP/IP address
Access to a specific TCP/IP address. If you know the TCP/IP address of the host, select this option and enter the address. This can be either of the following.

- An IPv4 dotted-decimal address (such as `193.1.11.100`).
- An IPv6 colon-hexadecimal address (such as `2001:0db8:0000:0000:0000:0000:1428:57ab` or `2001:db8::1428:57ab`).

TCP/IP port number
The TCP/IP port number that the TN Redirector uses to access the host.

Note: TCP/IP ports are completely unrelated to SNA ports.

You also need to configure this port number on the host. Port numbers above `2000` are likely to be available. Port numbers in the range `256–1023` may give slightly better security, but are more likely to be in use.

SSL secure session
Specify this option to indicate that TN Redirector uses Secure Sockets Layer (SSL) to access the host.

This option is available only if the host supports SSL.

Security level
Indicates the SSL security level required for the host session. The session will use the highest security level that both host and server can support; if the host cannot support the requested level of security or higher, the session will not be started.

This option appears only if you have selected the *SSL secure session* option.

Possible values are:

Authenticate Only
Certificates must be exchanged, but encryption will not be used. This option is typically used to avoid the overhead of encryption when the host connection is across a secure intranet.

Authenticate Minimum
The host must request a certificate from the server to check its validity; encryption is not required (but can be used if the host requests it).

40 Bit Minimum
The host must support at least 40–bit encryption.

56 Bit Minimum
The host must support at least 56–bit encryption.

128 Bit Minimum
The host must support at least 128–bit encryption.

168 Bit Minimum
The host must support at least 168–bit encryption.

Note: Using encryption requires additional software to be installed with Communications Server for Linux; see *Communications Server for Linux Quick Beginnings* for more information. Depending on your location, you may not be able to use all the encryption levels listed because the software required to support them is not available in your country.

Configuring SNA Gateway

Normally, a dependent LU session requires a direct communications link to the host computer. However, a node running Communications Server for Linux that has a direct communications link to the host can also provide SNA gateway facilities to LUs on downstream computers, enabling them to access the host over the communications link from the Communications Server for Linux node. The downstream computer must contain an SNA PU type 2.0 or 2.1 to support dependent communication with the host. For example, the downstream computer could be another computer running Communications Server for Linux in a standalone configuration.

Using the SNA gateway feature, all the data transferred between the host and the downstream computer is routed through the Communications Server for Linux local node. This enables a downstream computer to share a host connection with Communications Server for Linux or with other downstream computers, instead of requiring a direct link. For example, you can set up several downstream computers connected to Communications Server for Linux over a local token ring network, so that they all access the same long-distance SDLC leased line from Communications Server for Linux to the host.

Using SNA gateway also simplifies the configuration at the host. The host configuration needs to include only the Communications Server for Linux computer and its host communications link; the LUs at the downstream computers are configured as part of the resources of the Communications Server for Linux computer. The host computer is not aware that SNA gateway is being used.

Before configuring SNA gateway, you must perform the following configuration tasks:

- Define the local node as described in “Configuring the Node” on page 54.
- Configure a port and link station for dependent traffic between the local node and the host, as described in Chapter 4, “Defining Connectivity Components,” on page 59. Also, configure ports and link stations for dependent traffic between the local node and the downstream nodes. For downstream links, you can configure a template on the port to support implicit downstream LUs (LUs that are not explicitly defined on the local node).
- Define the LUs on the local node that are used for communication with the host (the upstream LUs). Upstream LUs, including Dependent LU 6.2 LUs, must be

defined using the LU Type 0–3 Configuration dialog, specifying an LU type of unrestricted (unknown). To add the LUs, see "Defining LU Types 0–3" on page 75.

- If you are going to use any LU pools, define them as described in "Defining LU Pools" on page 77.

To enable SNA gateway, you must configure LUs on the local node to support sessions with downstream workstations. (If you configured a template on the port to support implicit downstream LUs, you may not need to define downstream LUs explicitly.) The LUs defined on the local node are referred to as "downstream LUs." To configure downstream LUs, you need the LU numbers that are used on the downstream nodes, and the name of the host LU. (The LUs that are defined on the downstream nodes can be any dependent LU type.)

To configure downstream LUs, use one of the following methods:

Motif administration program
Select **SNA gateway** and **New downstream LU** from the **Services** menu on the Node window.

Command-line administration program
Issue one of the following commands:

define_downstream_lu

define_downstream_lu_range

Downstream LU Configuration Parameters

The following parameters are required for downstream LU configuration:

Downstream LU name
A name for each downstream LU. The LU name is used only to identify the LU locally, and does not need to match any configuration on the downstream node.

If you are defining a range of LUs, specify a base name of 1–5 characters. Communications Server for Linux adds a three-digit decimal string to the base name to create an LU name for each LU number you specify.

Downstream PU name
The name of the link station to the downstream node.

LU number
The LU number must match the LU number defined on the downstream node. Contact your SNA network planner if you do not know what LU number to use.

You can configure several LUs with consecutive LU numbers by defining a range of LUs.

Upstream LU name
The name of the host LU or a pool of LUs with which the downstream LUs will communicate.

Delayed logon
To reduce the user startup time, Communications Server for Linux displays a logon screen without assigning an upstream LU; a 3270 user must hit a key before the user is associated with an upstream LU.

Allow timeout
To reduce the number of LUs required, an LU without an active PLU-SLU session is disassociated from the upstream LU after this number of seconds.

Additional Configuration

After performing the downstream LUs for SNA gateway configuration, continue with the following configuration tasks:

- To configure user applications, see Chapter 7, "Configuring User Applications," on page 101.

Configuring DLUR

Normally, a dependent LU session requires a direct communications link to the host computer. If many nodes (including a host node) are connected together in an APPN network, some of them may have an indirect connection through another node instead of a direct connection to the host. Without a direct connection, it is not possible to establish dependent LU sessions to the host from LUs in these indirectly connected nodes.

Dependent LU requester (DLUR) is an APPN feature designed to overcome this limitation. DLUR can be configured on an APPN node (such as a node running Communications Server for Linux). It works in conjunction with dependent LU server (DLUS) at the host, to route sessions from dependent LUs on the DLUR node across the APPN network to the DLUS host.

The route to the host can span multiple nodes and can take advantage of APPN's network management, dynamic resource location, and route calculation facilities. DLUR must be available on the node where the LUs are defined, and DLUS must be available on the host node, but you do not have to enable DLUR on any intermediate nodes in the session route.

Note: You cannot configure DLUR on a LEN node.

If the Communications Server for Linux DLUR node is a network node, it can also provide passthrough DLUR facilities for dependent LUs on downstream computers connected to the Communications Server for Linux node. (Only network nodes support this function.) These downstream LUs can use DLUR on the Communications Server for Linux node to access the host across the network, in the same way that LUs internal to the node do.

Note: You cannot configure passthrough DLUR on an end node.

The tasks you need to perform to configure DLUR depend on whether the dependent LUs are on the local node or on downstream nodes.

To configure DLUR support on the local node, you must perform the following configuration tasks:

1. Define the local node as described in "Configuring the Node" on page 54. If you are providing passthrough DLUR support for downstream nodes, define the node as an APPN network node.
2. Configure connectivity to the APPN network. APPN connectivity requires at least a port and link station for independent traffic between the local node and the adjacent APPN network node, as described in Chapter 4, "Defining Connectivity Components," on page 59.

3. Define a DLUR PU on the local node as described in "Defining DLUR PUs" on page 71. (The DLUR PU supports connectivity to the host.)
4. To configure DLUR to support LUs on the local node, you must add the LUs on the local node, as described in Chapter 7, "Configuring User Applications," on page 101. The LUs can be configured to support 3270 display, 3270 printer, or LUA. Depending on the requirements of the user applications supported by the LUs, you may also need to perform further configuration.

To configure passthrough DLUR support for downstream nodes, you must perform the following configuration tasks:

1. Define the local node as an APPN network node (see "Configuring the Node" on page 54).
2. Configure connectivity to the downstream nodes. Configure ports and link stations for dependent traffic between the local node and each downstream node, as described in Chapter 4, "Defining Connectivity Components," on page 59. (You do not need to define a DLUR PU to support DLUR for downstream nodes.)
3. A downstream node can support multiple PUs. In this case, each downstream PU is associated with a different link, so you need to configure multiple links between the Communications Server for Linux DLUR node and the downstream node, and you need to know the downstream PU name for each link.

Chapter 9. Managing Communications Server for Linux from NetView

Communications Server for Linux includes a remote command facility (RCF) that operates in conjunction with the NetView program at a host computer, enabling a NetView operator to issue commands from the host NetView program to the Communications Server for Linux computer. (For a brief overview of NetView and RCF commands, see "Using the Host NetView Program.")

The Communications Server for Linux RCF provides the following two functions:

- Service point command facility (SPCF) enables a NetView operator to issue Communications Server for Linux administration commands from NetView using the same syntax as for the command-line administration program **snaadmin**. This facility is described in "Using SPCF" on page 118.
- UNIX command facility (UCF) enables a NetView operator to issue Linux operating system commands from NetView. This facility is described in "Using UCF" on page 119.

Both of these functions can be accessed from the NetView console in the same way, and the overall syntax for issuing the commands is the same.

Using the Host NetView Program

The Communications Server for Linux RCF operates in conjunction with the NetView program at a host computer. The host must be running NetView Version 1 Release 2, or a later version; Communications Server for Linux does not support NetView Version 1 Release 1.

To use the NetView program, you need the following:

- Login ID and password for the host NetView program (contact your host personnel for this information)
- Service point name for Communications Server for Linux, defined at the host for the NetView program (contact your host personnel for this information)
- DLC, port, and link station to access the host computer on which the NetView program is running

You may want to test the RCF function by using 3270 emulation to access NetView from Communications Server for Linux instead of accessing it directly from the host. In this case, you also require the following:

- 3270 LU configured at the host
- 3270 session using this LU

Consult your host administrator to obtain the necessary configuration information.

To access the NetView program, follow these steps:

1. Ensure that the Communications Server for Linux software is started, using a node configuration file that includes a definition of RCF access parameters (the `define_rcf_access` record).
2. If you are accessing the NetView program using 3270 emulation, start the 3270 emulation program and activate the session to the host.

3. Follow the instructions given to you by the host administrator for starting NetView and logging on. (The sequence of operations may vary with different versions of NetView.)
4. Issue SPCF or UCF commands as required.
5. If you are using 3270 emulation to access NetView, follow the instructions in your 3270 documentation for ending 3270 emulation when you have finished issuing commands.

NetView Screen Display

The layout of the NetView screen varies with different versions of NetView at different hosts. A typical layout is shown in Figure 19.

The display includes an input area at the bottom of the screen; this is the area into which you can type commands. The line `???` divides the main screen area (where NetView displays responses to your commands) from the input area.

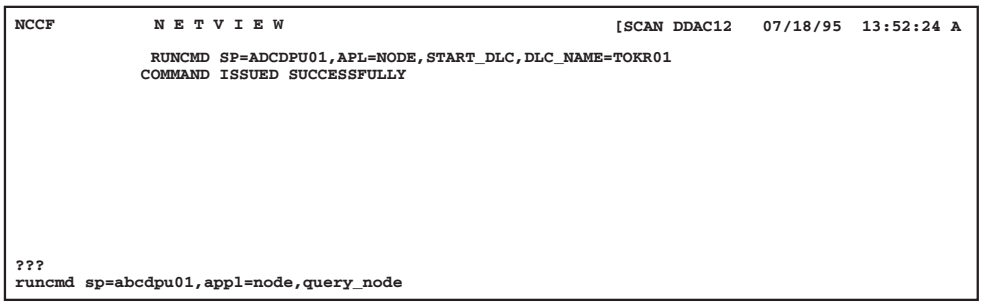

Figure 19. Example of a NetView Screen

Changing the Size of the Command Input Area

By default, the input area is one line, but for some of the longer commands you need more than one line. On some versions of NetView, you can specify an input area of one, two, or three lines by using the **input** command. To do this, type the following command:

input *n*

In this command, *n* is 1, 2, or 3, indicating the number of lines you want. If this command does not work on the version of NetView you are using, contact your NetView support personnel.

Overview of RCF Command Syntax

Both SPCF and UCF commands use the RCF command syntax:

runcmd sp=*spname*, **appl=***component*, *commandtext*

NetView uses the **runcmd** utility to send a command string to a remote system. The command includes the following parameters:

sp=spname
: Indicate the service point name (defined at NetView) that corresponds to the Communications Server for Linux node. The host NetView personnel can give you this information.

appl=component
Indicate the name of the Communications Server for Linux component to which NetView should send the command, as follows:

node The Communications Server for Linux node associated with the service point name *spname* (for SPCF commands)

unix The UCF daemon program running on the Communications Server for Linux computer associated with the service point name *spname* (for UCF commands)

commandtext
Supplies the text of the command being issued. For SPCF, this is a command issued to the Communications Server for Linux command-line administration program. For UCF, it is a command for the Linux operating system. For more information about the commands that can be used, see "Restrictions on Administration Commands Used with SPCF" on page 118 or "Permitted Commands" on page 120.

Uppercase Characters and Escape Characters

Although Linux distinguishes between uppercase and lowercase alphabetic characters, the NetView program may not do so. The NetView **netvasis** command can be used to provide mixed-case input to **runcmd**, but Communications Server for Linux RCF has no way to determine whether **netvasis** is in use. Because RCF cannot determine whether an alphabetic character received from the host was originally uppercase or lowercase, it assumes that received characters are intended to be lowercase. Also, the host character set may not support the square bracket characters [and], which are required in some commands.

RCF provides support for uppercase characters and square bracket characters using the backslash character \, as follows:

- To include an uppercase character in the command string, include a backslash character before it. Any alphabetic character not preceded by a backslash is interpreted as lowercase.
- To include the square bracket characters [and], use the sequences \(and \), respectively.
- To include the backslash character \ itself, type it twice.

If a single backslash is followed by any other nonalphabetic character, the backslash is ignored and the character is left unchanged.

Some examples are shown in Table 3.

Table 3. Using Escape Characters in RCF Commands

Characters to Produce	Input
ABcd	**\a\bcd**
[ ]	**\(\)**
\a	**\\a**
\[	**\\\(**

The escape characters you would normally use on the Linux command line, to prevent the Linux shell from interpreting special characters, are not required with RCF. For example, do not use escape characters with strings containing the characters * or $, as you would when entering them on the Linux command line.

Also, when using SPCF to issue administration commands, be aware that constant names such as LIST_FROM_NEXT are not case-sensitive. You do not need to escape these characters to make them uppercase.

Using SPCF

SPCF enables you to issue commands from the NetView console to manage the running Communications Server for Linux system. These commands are the same as those you can issue using the Communications Server for Linux command-line management program **snaadmin** (as described in *Communications Server for Linux Administration Command Reference*).

For information about the syntax of an SPCF command, see "Overview of RCF Command Syntax" on page 116. The command text following the *appl*=node parameter is a command issued to the Communications Server for Linux command-line administration program, in the same format as you would specify it to the **snaadmin** program on the Linux command line. Refer to *Communications Server for Linux Administration Command Reference* for information about the syntax of administration commands and the parameters for individual commands.

Restrictions on Administration Commands Used with SPCF

You cannot use the command-line option **-i** to specify input from a file or from standard input. All commands must be entered directly at the NetView console.

With **query**_* commands, you can use the command-line options **-a** (return all entries) and **-d** (return detailed information) in the same way as when entering commands on the Linux command line.

To provide security, you can set up the Communications Server for Linux configuration so that only certain types of commands are permitted from SPCF. For example, you can permit remote users to issue **query**_* commands, but not to activate or deactivate Communications Server for Linux components. You can control access separately for each of the following groups of commands:

- **define**_*, **set**_*, **delete**_*, **add**_*, and **remove**_* commands, and also **init_node**
- **query**_* commands
- "Action" commands: **start**_*, **stop**_*, **activate**_*, **deactivate**_*, and also **aping**, **initialize_session_limit**, **change_session_limit**, and **reset_session_limit**

For more information about setting up security options for SPCF, refer to the description of the **define_rcf_access** command in *Communications Server for Linux Administration Command Reference*.

Examples of SPCF Commands

The following example shows how you could issue the **define_lu_0_to_3** command using SPCF. This example uses backslash characters to indicate uppercase letters in the two character strings LU$01 and PU2. There is no need to make the characters in the constant name 3270_display_model_2 uppercase, because the **snaadmin** program accepts this string in lowercase.

```
runcmd sp=myspname, appl=node, define_lu_0_to_3, lu_name=\l\u$01,

        nau_address=1, pu_name=\p\u2, lu_model=3270_display_model_2
```

The following example shows how you could issue the **query_lu_0_to_3** command using SPCF. The **-a** option indicates "return all entries," so there is no need to specify an LU name or PU name. The **-d** option indicates "return detailed information," so there is no need to specify this using the *list_options* parameter. These two options act in exactly the same way as for the **snaadmin** program.

runcmd sp=myspname, appl=node, -a -d query_lu_0_to_3

Using UCF

UCF enables a NetView operator to issue Linux commands on a computer running Communications Server for Linux by typing the command text at the NetView console, and to view output from these commands. The facility is not restricted to commands related to Communications Server for Linux; subject to the restrictions in "Permitted Commands" on page 120, any type of command can be issued.

By using UCF, a remote operator can monitor activity on the Communications Server for Linux computer, diagnose problems, and in some cases take corrective action.

You can specify whether Communications Server for Linux supports UCF by using the **define_rcf_access** command (refer to *Communications Server for Linux Administration Command Reference*). If the configuration specifies that UCF is supported, Communications Server for Linux starts the UCF daemon program when the node is started. The UCF daemon processes Linux commands from the UCF by starting a new Linux shell for each command and running the command in that shell. If UCF support is not included, Communications Server for Linux does not start this program.

The configuration specifies the name of the UCF user, which must be a valid login name on the Communications Server for Linux computer. The UCF shell is started using the shell program, login ID, permissions, and **.login** or **.profile** specified for that user. (If no shell program is specified, **/bin/sh** is used.) This means that the normal Linux system security features can be used to restrict the UCF user's access to files and commands, and therefore to limit the range of commands available from UCF.

For more information about setting up the UCF configuration, refer to the description of the **define_rcf_access** command in *Communications Server for Linux Administration Command Reference*.

UCF Command Syntax

The syntax of a UCF command is as follows:

runcmd sp=*spname***, appl=unix,** *unix_command*

NetView uses the **runcmd** utility to send a command to a remote system. The command includes the following parameters:

*sp***=***spname*
: Specify *spname*, which is the name of your service point as defined at NetView. The host NetView personnel can give you this information.

appl=**unix**
: Instruct NetView to send the command to the UCF daemon program on the Communications Server for Linux computer associated with the service point name *spname*.

unix_command
: Supply the Linux operating system command. This command is entered as you would enter it on the Linux command line, except for the escape characters to indicate uppercase letters or square bracket characters (as described in "Overview of RCF Command Syntax" on page 116).

  The escape characters you would normally use on the Linux command line, to prevent the Linux shell from interpreting special characters, are not required with UCF. For example, do not use escape characters with strings containing the characters * or $, as you would when entering them on the Linux command line.

Permitted Commands

The UCF is designed for use with commands that complete (whether or not any output is produced) without any further interaction from the user. For example, you can issue the command **cat** *filename*, which completes after displaying the contents of *filename*, or **mv** *filename1 filename2*, which completes with no output unless an error occurs.

Output generated by a UCF command is returned to UCF when the Linux operating system command completes. This leads to the following restrictions:

- Any output generated after the command completes is not returned to UCF. For example, if you issue a command followed by & to run it in the background, UCF receives the operating system message giving the process ID of the background command, but does not receive any subsequent output that is generated. Similarly, you can use UCF to start a daemon process, but you cannot see any output generated by the process.
- The UCF cannot be used with a command that requires further input from the user before it completes (for example, a command such as **vi** *filename* that starts an interactive process, or a command such as **tail -f** *filename* that does not complete until it is stopped by the user).

Because all Linux commands run with the login ID and permissions of the configured UCF user, the valid commands are limited by the access rights of the UCF user's login. In particular, root or superuser commands are not permitted. For more information, see "UCF Security" on page 122.

Example of a UCF Command

The following is an example of a UCF command as you would enter it from NetView:

runcmd sp=myspname, appl=unix, grep \temp \(ab\)*.c >\t\e\m\p.out

The command that would run on the Linux computer is:

grep Temp [ab]*.c >TEMP.out

Output from Linux System Commands

When a command is issued successfully, the following messages are displayed on the NetView screen:

= = = EXECUTING UNIX COMMAND = = =
(any output from the command, including error messages)
= = = UNIX COMMAND COMPLETED = = =

These messages may not appear on the NetView screen at the same time. The **EXECUTING UNIX COMMAND** message appears as soon as the UCF daemon program receives the command and returns control to the NetView operator. Any output from the command is sent to NetView as it is produced, and may appear as a series of separate messages; the **UNIX COMMAND COMPLETED** message appears when the Linux command has finished and its shell has ended.

If the output from the Linux command contains tab characters, Communications Server for Linux converts each tab to a space character before sending the output to NetView. Otherwise the output is sent unchanged.

If you issue a command when a previous command is still in progress (that is, before the **UNIX COMMAND COMPLETED** message is received), the following message is displayed:

= = = COMMAND QUEUED = = =

The second command is queued, and is executed when the previous command has completed.

Canceling a Command

UCF provides a method of canceling a command that is still in progress. This can be used to stop the current command from executing, or to cancel an interactive command such as **vi** *filename* that cannot complete without further input. It is equivalent to using an interrupt sequence such as **Ctrl + C** to stop a process running on a terminal, or using the Linux **kill** command to stop the process.

In addition to canceling the command that is currently executing, Communications Server for Linux cancels any commands that are queued after it.

The command syntax is the same as for the Linux command, with the string **ux-cancel** instead of the command text. For example:

runcmd sp=myspname, appl=unix, ux-cancel

For each outstanding command (the one currently executing and any queued commands), the following message is displayed:

= = = UNIX COMMAND CANCELLED = = =

This message indicates that the Linux shell in which the command was running has been stopped. Further Linux commands can be issued as necessary.

If a command starts a daemon process on the Linux computer, this process may not be stopped by **ux-cancel**. You may need to use the Linux **kill** command (either on a terminal or by using UCF) to stop such a process explicitly.

If no UCF command is running when **ux-cancel** is used, UCF displays the following message:

NO OUTSTANDING COMMANDS

In this case, the **ux-cancel** command is ignored. No action is necessary. This message can be displayed when the **ux-cancel** command is issued after the previous command finishes but before the **UNIX COMMAND COMPLETED** message is received.

UCF Security

Because the UCF enables a remote operator to issue commands on the Linux computer and to receive output from these commands, it is important to consider the security implications. For example, you need to ensure that the operator cannot access private information or issue Linux commands that can disrupt other users.

The Communications Server for Linux configuration includes a specific Linux system user name as the UCF user; this must be a valid login ID on the Communications Server for Linux computer. All UCF commands run with this user's ID, and therefore with the access permissions of this user.

It is intended that you use the normal security features provided by Linux to restrict the commands the UCF user can access, in order to permit only those commands you consider reasonable for use from UCF. The following guidelines may be useful:

- The UCF user name should be one that is used solely for UCF; you should not use an existing login that is also used for other purposes. This makes it easier to define the privileges of this user to include only those that are reasonable for UCF; it also enables you to identify processes that were started using UCF.
- You may need to restrict the users and groups for which the UCF user can change a user ID or group ID. In particular, the UCF user must not be permitted to do the following:
  - Become root or superuser.
  - Use the group ID `system`, which enables access to the **snaadmin** program. (The functions of this program should be accessed using SPCF, as described earlier in this chapter, instead of UCF.)

Chapter 10. Managing Communications Server for Linux Client/Server Systems

Communications Server for Linux can operate as a standalone system with all SNA components and applications on a single Linux system, or can operate as part of a client/server domain. A client/server domain includes both servers (SNA nodes) and IBM Remote API Clients (which can access SNA connectivity through a server).

In a domain with multiple Communications Server for Linux servers, one server holds the master copy of the Communications Server for Linux domain configuration file. This server is known as the master server. You can define other servers in the domain to be backup servers. The domain configuration file is copied to backup servers—either when they are started, or when the master copy is changed—so that all backup servers hold a copy of the latest information.

Remote API Clients can be computers running AIX, Linux, Linux for pSeries, Linux for System z, or Microsoft® Windows.

Servers and clients communicate across the Communications Server for Linux domain using TCP/IP; both IPv4 and IPv6 addressing are supported. A client can access one or more servers at the same time, and can run concurrent applications as needed. For information about the networking requirements for a client/server configuration, see "IP Networking Requirements" on page 125.

The TCP/IP connections used between clients and servers may flow across physical LANS, WANs, or virtual paths between servers running under VM. In the Communications Server for Linux books, the term LAN is used for all of these.

UNIX

For Remote API Clients on AIX or Linux, you must supply information about the Communications Server for Linux network and servers. For information about this function, and for instructions on enabling and disabling the Communications Server for Linux software on clients, see "Managing Remote API Clients on AIX or Linux" on page 142.

All administration commands can be issued on a server. However, there are restrictions on which commands can be issued on AIX and Linux clients.

- You can issue any **query** or **status** command on a an AIX or Linux client.
- Some other administration commands, defined in Communications Server for Linux Administration Command Reference, explicitly say that they can be issued from an IBM Remote API Client. Otherwise these commands are available only from a server.

WINDOWS

For Windows clients, you must supply information that Communications Server for Linux can use to enable the client software. If you plan to have invokable TPs

on the Windows client, you must also supply information about the TPs. For information about these functions, and for instructions on enabling and disabling the Communications Server for Linux software on a Windows client, see "Managing Remote API Clients on Windows" on page 128.

Administration commands, defined in Communications Server for Linux Administration Command Reference, cannot be issued from a Windows client.

Changing Client/Server Configuration

When you install the Communications Server for Linux software, as described in *Communications Server for Linux Quick Beginnings*, it is initially installed in standalone mode (with all components on a single Linux computer). If you want to run Communications Server for Linux as a client/server system, you can then configure one server to be the master server, and configure any other servers as backup servers. (You are recommended to configure all servers other than the master as backup servers.)

Communications Server for Linux includes a command-line application program, **snanetutil**, to make a server part of a client/server domain. To do this, use the following command on each server (starting with the master server):

sna stop
snanetutil *master_name* **[***domain_name***]**
sna start

The parameters in the **snanetutil** command are as follows.

master_name
The name of the master server in the domain to which the server belongs. If you are moving the server into an existing domain, this must match the name of the existing master server in that domain.

domain_name
The name of the domain to which the server belongs. This parameter is optional; if you do not specify it, Communications Server for Linux uses the default domain name `ibmcs_domain`.

To configure each server other than the master as a backup server, issue the following command. You can do this on the backup server itself or on the master server, but the Communications Server for Linux software must be running on the master server in either case.

snaadmin add_backup, backup_name=*server_name*

server_name is the name of the server that you want to add as a backup server.

You can also use the **snanetutil** program to move the server out of an existing domain so that it runs as a standalone system.

Note: Do not use this option unless you want to stop running Communications Server for Linux as a client/server system and use it simply as a standalone node. If you remove all the servers from an existing domain, any clients left in that domain will be unable to access SNA resources.

To move a server out of its domain so that it runs as a standalone system, use the following command:

snanetutil -d

Moving Clients Into a Different Domain

The **snanetutil** program allows you to move servers between different client/server domains. If you want to move clients between domains, you need to do this by modifying the client configuration.

On each Remote API Client on Windows that is to be moved, use the Client Configuration Utility to change the *domain* parameter to match the new domain name. See "Remote API Client on Windows Configuration" on page 130 for more information.

On each Remote API Client on AIX or Linux that is to be moved, change the *domain* entry in the `Configuration` section of the client network data file to match the new domain name. See "Client Network Data File (sna_clnt.net)" on page 143 for more information.

IP Networking Requirements

Remote API Clients can communicate with Communications Server for Linux servers using TCP/IP, or using HTTPS via a WebSphere® server. See "HTTPS Access for Remote API Clients" on page 128 for more details about using HTTPS connections.

Before you can run the Remote API Client, you must configure TCP/IP port addresses on both the clients and servers in your network. If you encounter problems with the default port assignments, you may need to resolve conflicts as described in "Setting Up IP Port Numbers" on page 126.

In addition, you may wish to set clients up so that the TCP/IP connection is dropped automatically when the client is finished using Communications Server for Linux, as described in "LAN Access Timeout" on page 127.

IPv4 and IPv6 Addressing

Computers in a Communications Server for Linux client/server domain can use either IPv4 or IPv6 addresses, but all servers in the domain must use the same addressing format (IPv4 or IPv6).

- If the servers use IPv4, clients must also use IPv4.
- If the servers use IPv6, clients can use either IPv6 or IPv4.

For more details of how to set up and use IPv4 and IPv6 addressing, see *Communications Server for Linux Quick Beginnings*.

Host Names in Client/Server Configuration

Communications Server for Linux uses fully-qualified IP hostnames for its internal communications between servers and clients. Normally the local system can determine these names from network configuration (such as DNS). If this is not possible, you should use a fully-qualified name (such as `newbox.this.co.uk`)rather than an alias (such as `newbox`) whenever a hostname is required in configuration.

The local server name for each computer is taken from the **/etc/hosts** file. The entry in this file must specify the IP address first, then the fully-qualified name, and finally the alias, for example:

```
9.42.108.28    newbox.this.co.uk    newbox
```

If the server is multi-homed (for example, if it has two or more TCP/IP network interfaces so that it appears with different IP addresses), the entries in the **/etc/hosts** file must specify the same IP name for all addresses, so that the name can be resolved correctly for all network interfaces. For example:

```
9.42.108.28    newbox.this.co.uk    newbox
9.42.80.127    newbox.this.co.uk    newbox
```

You also need to include the line `multi on` in the **/etc/host.conf** file to indicate that the server is multi-homed.

Setting Up IP Port Numbers

Communications Server for Linux uses both TCP/IP and UDP/IP communications to send client/server data across the LAN. By default, it uses the port number 1553 for both types of communications. For most installations, this port number should be suitable; you do not need to change it.

If you encounter problems enabling the Communications Server for Linux software, check the error log file for messages indicating that the port number used by Communications Server for Linux clashes with the port number used by another program. If you find such messages, take the following steps:

1. Check the **/etc/services** file on the computer where the error occurred, to see if another program is listed as using the port number 1553 for either TCP/IP or UDP/IP communications. If this is the case, first try to change the other program to use a different port.
2. If you cannot do this, or if no program is listed as using port 1553, find another port number that is not listed in the file as being used by any program. Check the **/etc/services** file on all other Communications Server for Linux computers in the same domain, to ensure that the number is not used on any other computer.
3. In the **/etc/services** file on each computer in the domain, add two lines in the following form:

   ```
   sna-cs          nnnn/tcp
   sna-cs          nnnn/udp
   ```

   The *nnnn* entry is the new port number. This must be set to the same value on all computers in the Communications Server for Linux domain.
4. WINDOWS

   If your Communications Server for Linux domain includes Windows clients, add the same two lines to the **services** file on each Windows computer. The **services** file is in the same format as the Linux file, and is generally stored in the home directory of the Windows TCP/IP software; see your Windows TCP/IP documentation for more information if necessary.

   
5. Re-enable the Communications Server for Linux server and Remote API Client software.

Note: You are advised to use a firewall to protect port 1553 on the server (or the new port number you have specified for client/server communications), to prevent unauthorized access. Both TCP and UDP traffic should be permitted to and from other Communications Server for Linux servers and Remote API Clients, but no other computers should be permitted to access the port.

LAN Access Timeout

If the client is communicating with Communications Server for Linux servers across a network for which connection charges are payable, you may want to ensure that the TCP/IP connection from the client is dropped automatically after applications on the client have stopped using Communications Server for Linux resources. This does not automatically disable the SNA software on the client; it remains active, and attempts to re-establish contact with a server if an application requires it at a later time.

The *lan_access_timeout* parameter (in the **sna_clnt.net** file for a Remote API Client on AIX or Linux, or the Registry for a Remote API Client on Windows) enables you to disable the SNA software on the client. The TCP/IP connection is dropped when none of the following events have occurred on the client for the specified time:

- APPC or CPI-C conversations active (or attempts to start a conversation)
- LUA sessions active
- CSV TRANSFER_MS_DATA verbsfrom a Windows client
- MS verbs (Linux clients only)
- NOF verbs (except the **query_central_logger** or **query_node_all** verbs)
- Administration commands (except the following events, which do **not** cause the client to restart the connection):
  - Error or audit messages logged by the client (these are logged locally on the client, even if central logging is being used)
  - The administration commands **query_central_logger** or **query_node_all** (these return the information that was available before the TCP/IP connection was dropped, and so may not match the current status of the LAN)
  - The NOF verbs **query_central_logger** or **query_node_all** (as for the equivalent administration commands)

In particular, the TCP/IP connection is dropped if you enable the SNA software but do not start any Communications Server for Linux applications on the client within the specified timeout.

When one of these events occurs while the TCP/IP connection is down, the client re-starts the attempt to contact a server, as described for the * and *servername* parameters in “Client Network Data File (sna_clnt.net)” on page 143, or “Servers” on page 134.

Incoming Attaches for invoked TPs on this client cannot be accepted while the TCP/IP connection is down; the Attach is rejected as though the target system were inactive. This means that automatically started TPs on the client are not available if no other applications on the client are running and the TCP/IP connection has timed out. However, operator-started TPs on the client can be used at any time, because a Receive_Allocate verb issued by the TP re-establishes the TCP/IP connection.

HTTPS Access for Remote API Clients

If you are running a client/server system in which Remote API Clients connect to Communications Server for Linux servers using HTTPS, you will need a computer running WebSphere Application Server to provide HTTPS access from these clients to the servers. For instructions on how to install and configure this server, see *Communications Server for Linux Quick Beginnings*.

If you add new servers to the Communications Server for Linux domain and you want Remote API Clients to be able to access these servers using HTTPS, you will need to update the WebSphere server configuration file to include these servers. This file is named **snahttpsrv.cfg**, and is stored on the WebSphere server in the directory specified by the **USER_INSTALL_ROOT** environment variable. If you are not sure where this is, take the following steps.

1. Start the WebSphere administration console.
2. In the administration console menu bar, choose Environment, Manage WebSphere Variables.
3. Look for the **USER_INSTALL_ROOT** variable in this list, and note its value (which is the path of a directory on the WebSphere server). The list of environment variables may span two or more pages, so you may need to use the Next button to scroll through the list.

Edit the configuration file using a text editor to include a list of all Communications Server for Linux servers that can be accessed by Remote API Clients using HTTPS. Each server must be specified on a separate line of the file, in the following format:

server=*servername.domainname.com*

On each Remote API Client that will access the new server, you also need to add the new server name to the list of servers in the client network data file (or in the Windows Registry for a Windows Client). See the section for the appropriate client type later in this chapter.

Managing Remote API Clients on Windows

WINDOWS

Communications Server for Linux enables machines running Microsoft Windows to act as clients in the Communications Server for Linux domain. The Communications Server for Linux client software includes API libraries that are compatible with Microsoft Host Integration Server, the Windows Open Systems Architecture (WOSA), and the interfaces provided by IBM Personal Communications and Communications Server for Windows. This enables applications written for these implementations to run unchanged on the Remote API Client on Windows.

The Remote API Client on Windows supports the following WOSA APIs:

- Windows APPC
- Windows CPI-C
- Windows LUA
- Windows CSV

For more information about Windows SNA APIs, see the documentation provided with Microsoft Host Integration Server.

SNA network information, and other information required by the Remote API Client on Windows, is held in the Windows Registry.

The client must be enabled before you can use Communications Server for Linux applications or emulation programs on the client. For more information, see "Enabling a Remote API Client on Windows." When the client is enabled, it contacts a server running Communications Server for Linux over the TCP/IP network in order to access Communications Server for Linux features.

The operation of the client is also controlled by the information in the Windows Registry. The Windows Registry contains information about the following:

- Configuration information specific to Remote API Clients on Windows
- Servers that the client can access
- Logging and tracing options for applications running on the client
- Additional options for CPI-C and CSV applications running on the client
- Invokable TPs (APPC or CPI-C) that can run on the client

The most commonly used parameters can also be modified using the Client Configuration Utility, which is the preferred method for modifying them. For more information, see "Remote API Client on Windows Configuration" on page 130.

Note: If the client uses HTTPS to access its servers, you will need to modify the client configuration to specify the names of these servers and the WebSphere server providing HTTPS access to them before you can use the client. See "Remote API Client on Windows Configuration" on page 130 for more information.

Enabling a Remote API Client on Windows

The Remote API Client on Windows runs as a Windows service. The installation program configures it to start automatically when the computer starts. If necessary, you can start it manually in either of the following ways.

- Start the client service from the Services applet under Control Panel, Administrative Tools.
- Type `net start sxclient` from a command window or from the Start / Run icon.

The client then uses the information in the Windows Registry, defined using the Client Configuration Utility and described in "Remote API Client on Windows Configuration" on page 130, to locate a server running Communications Server for Linux.

Note: If you are using Microsoft Windows Vista, you need to run `net start sxclient` from a command prompt that has administrator authority. To access this command prompt, use the right-hand mouse button on the Command Prompt icon, select 'Run as administrator', and type the administrator password at the prompt.

Viewing Status of a Remote API Client on Windows

The Client Monitor places an icon in the system tray that displays the Client's status when you move the mouse pointer over it. The Monitor is set up to run

automatically when the computer starts, but you can also run it manually from the Start menu if necessary. The status is one of the following:

Not Active
The client has not been started.

Not Connected
The client has been started, but has not yet made contact with a server (or has lost contact).

Server_Name
The client is connected to the named server.

Disabling a Remote API Client on Windows

Before disabling the client, ensure that all Communications Server for Linux applications (3270 and 5250 emulation programs, or applications using the Communications Server for Linux APIs) on the client have been stopped.

To disable the client, stop the Client service in one of the following ways.

- Stop the Client service from the Services applet under Control Panel, Administrative Tools.
- Type `net stop sxclient` from a command window or from the Start / Run icon.

On a computer running Windows Terminal Services, this means that all users are prevented from using the client.

Note: If you are using Microsoft Windows Vista, you need to run `net stop sxclient` from a command prompt that has administrator authority. To access this command prompt, use the right-hand mouse button on the Command Prompt icon, select 'Run as administrator', and type the administrator password at the prompt.

Remote API Client on Windows Configuration

On Remote API Clients on Windows, configuration information is held in the Windows Registry. The Registry contains SNA network information(similar to the information held in the client network data file on Remote API Clients on AIX or Linux). It also contains some additional configuration information that is specific to Remote API Clients on Windows.

Note: Configuration information for a CPI-C application (the local TP name and local LU alias) can be specified either in environment variables or in the registry. You may need to use environment variables if you are using Windows Terminal Server and need to run multiple copies of the same application using different local LUs. For more details, see "Appl_Name" on page 141.

The Client Configuration Utility provides a simple way of modifying the most commonly used client configuration parameters, and is the preferred method for modifying these parameters. To use this program, run the **Configuration Utility** program, located in the IBM Remote API program group. The program displays the same Configuration window that was displayed in the initial install process. Refer to the Remote API Client on Windows installation chapter in Communications Server for Linux Quick Beginningsfor more details of how to modify these configuration parameters.

Note: After changing the client configuration parameters, you need to stop and restart the client before your changes take effect. For details of how to do this, see "Disabling a Remote API Client on Windows" on page 130 and "Enabling a Remote API Client on Windows" on page 129.

In the Registry, the information is contained in values configured under subkeys of the following key:

```
\\HKEY_LOCAL_MACHINE\SOFTWARE\SNA Client\SxClient\Parameters
```

The possible values for each Registry subkey are as follows:

```
Configuration
   domain = domain_name
   maximum_process_count = nn
   maximum_header_count = nn
   maximum_element_count = nn
   invoked_tps = YES | NO
   lan_access_timeout = nn
   broadcast_attempt_count = nn
   server_lost_timeout = nn
   client_start_timeout = nn

Servers
   Server1 = * | [ webservername : [ portnumber : ] ]servername1
   Server2 = [ webservername : [ portnumber : ] ]servername2
      .
      .
      .
   Server9 = [ webservername : [ portnumber : ] ]servername9
Logging
   exception_logging_enabled = YES | NO
   audit_logging_enabled = YES | NO
   log_directory = directory
   error_file = error_filename
   backup_error_file = backup_error_filename
   error_file_wrap_size = error_file_size
   audit_file = audit_filename
   backup_audit_file = backup_audit_filename
   audit_file_wrap_size = audit_file_size
   succinct_errors = YES | NO
   succinct_audits = YES | NO

API_tracing
   file1 = trace_filename_1
   file2 = trace_filename_2
   flip_size = filesize
   truncation_length = length
   all_api = YES | NO
   appc = YES | NO
   cpic = YES | NO
   csv = YES | NO
   rui = YES | NO
   nof = YES | NO

CS_tracing
   file1 = cs_trace_filename_1
   file2 = cs_trace_filename_2
   flip_size = filesize
   admin_msg = YES | NO
   datagram = YES | NO
   data = YES | NO
   send = YES | NO
   receive = YES | NO

Internal_tracing
   file1 = internal_trace_filename_1
```

```
   file2 = internal_trace_filename_2
   flip_size = filesize
   trace_level = nn
   trace_flushing = YES | NO

Appl_Name
   APPCTPN = tp_name
   APPCLLU = lu_name

CSV_data
   CSVTBLG = table_G_filename
```

Note: The `domain = domain_name` value is the only required value in the Registry.

The following sections explain the configuration parameters. Where a parameter takes the values `YES` or `NO`, any string beginning with `Y` or `y` is interpreted as `YES`, and any string beginning with `N` or `n` is interpreted as `NO`.

Configuration

The `Configuration` subkey contains configuration information for the client, as follows:

domain The Registry data type of this value is `REG_SZ`.

The *domain_name* value indicates the domain name of the Communications Server for Linux LAN, as specified during the client installation. This line is required.

maximum_process_count

The Registry data type of this value is `REG_SZ`.

Specify the maximum total number of APPC, CPI-C, LUA and NOF applications that can run on this client at any one time.

This parameter is optional; the default value is `240`, which should normally be sufficient. If you see error messages reporting a failure to allocate an IPC control block, you may need to increase the maximum process count by specifying this parameter; the largest value you can specify is `4096`.

maximum_header_count, *maximum_element_count*

The Registry data type of these values is `REG_SZ`.

These two parameters are optional; the default values are `1250` and `1800`, which should normally be sufficient. You will not normally need to supply values for these parameters except as instructed by support personnel.

invoked_tps

The Registry data type of this value is `REG_SZ`.

Specify one of the following values:

YES This client is used to run invoked TPs (APPC TPs that issue RECEIVE_ALLOCATE, or CPI-C applications that issue Accept_Conversation or Accept_Incoming). In this case, you may also need to define the TP on this client. For more information, see "Defining TPs" on page 86 or Appendix B, "Configuring an Invokable TP from the Command Line," on page 171.

NO This client is not used to run invoked TPs.

This line is optional. If it is not specified, the default is `NO`.

lan_access_timeout

The Registry data type of this value is `REG_SZ`.

Specify the time in seconds for which the IP or HTTPS connection from the client to a server should be kept active while no applications on the client are using Communications Server for Linux resources. For more information, see "LAN Access Timeout" on page 127.

The valid range is 0–65535. The minimum timeout is 60 seconds (lower values are rounded up to 60 seconds). To deactivate the connection more quickly, disable the client.

This parameter is optional. If it is not specified, the default is no timeout, and the connection is kept active as long as the client is running.

broadcast_attempt_count

The Registry data type of this value is REG_SZ.

If the client uses the broadcast method to contact a server (specified by the * entry described in "Servers" on page 134), this parameter specifies the maximum number of broadcasts to be made in one attempt to contact a server.

The valid range is 1–65535. The minimum value is 1; if a higher value is specified, the client retries every 10 seconds until it contacts a server or until this count is reached. If the count is reached without contacting a server, the client then attempts to contact a named server (as described in "Servers" on page 134).

This parameter is optional. If it is not specified, the default is 5.

server_lost_timeout

The Registry data type of this value is REG_SZ.

If the client loses contact with a server and needs to reconnect, or if it has failed to contact a server using either broadcasts or named servers (as described in "Servers" on page 134), this parameter specifies the time in seconds for which the client waits before attempting to contact a server. If the client has lost contact with the server, Communications Server for Linux does not wait for the full timeout period, but retries after a random period between 5 seconds and the specified timeout; this is to avoid bursts of network traffic caused by large numbers of clients attempting to contact a server at the same time.

This parameter is optional. The valid range is 5–65535. If it is not specified, the default is 200 (seconds).

client_start_timeout

The Registry data type of this value is REG_DWORD.

Specify the time in seconds that an application waits while the client starts and tries to contact a server. Values between 0 and 300 are valid; values outside this range are forced into the range. The default value is 10 seconds.

This parameter can be used to control events when both the application and the client are configured to be started on system startup (either by being in the Startup Folder or by being an automatically started service). The application waits for the number of seconds specified in this field, to enable the client to get in first. In this way, the client can connect to a server to provide the resources required by the application, before the application fails due to the lack of those resources.

Servers

The Servers subkey contains information about Communications Server for Linux configuration servers that the client should contact to obtain configuration information. This list should contain the names of the master configuration server and any backup servers in the same domain as the client. For information about configuring master and backup servers, see "Configuring Client/Server Functions" on page 53.

Note: The format and meaning of this subkey depends on whether the client is on the same private network as its servers or connecting across a public network using HTTPS, as noted below.

Server1

The Registry data type of this value is REG_SZ.

Enter an asterisk (*) or a server name:

- To indicate that the client is on the same private network as its servers and should attempt to find a server running Communications Server for Linux by using a UDP broadcast message to all computers on its TCP/IP subnet (or on all subnets that it can access, if the client computer contains more than one LAN adapter card), specify *.

  This option is available only if the client uses IPv4 addressing. UDP broadcasts are not supported for IPv6.

  The client retries the broadcast every 10 seconds, up to the number of attempts specified by the *broadcast_attempt_count* parameter, until it contacts a server. If the limit specified by *broadcast_attempt_count* is reached before a server has been contacted, the client then tries using directed messages to one or more named servers (specified by the following lines of the file).
- In situations where the client is on the same private network as its servers but cannot reach any servers using UDP broadcasts, and must use directed messages, specify the name of the first server it should try to contact. The *webservername* and *portnumber* parameters are not used and should not be specified. This applies in the following cases:
  - When the client uses IPv6 addressing (UDP broadcasts are not supported for IPv6).
  - When the Communications Server for Linux LAN spans multiple TCP/IP subnets, and there are no Communications Server for Linux servers in any TCP/IP subnet that the client can access using UDP.
  - When UDP support is not installed on the client.

  In other cases, the use of UDP broadcasts is optional; to specify that broadcasts should not be attempted, specify the name of the first server instead of *.
- If the client uses HTTPS to access its servers, UDP broadcasts are not supported. In this case, specify the name of the WebSphere server that provides HTTPS support and the name of the Communications Server for Linux server, in the following format:

  webservername **:** *servername1*

  This assumes that WebSphere is set up to use the default port 443 for HTTPS connections. If your network administrator has configured WebSphere to use a different port number, include the port number in the following format:

  webservername **:** *portnumber* **:** *servername1*

For more details about configuring WebSphere to support HTTPS connections, refer to *Communications Server for Linux Quick Beginnings*.

Server2–Server9

The Registry data type of this value is REG_SZ.

Specify the names of additional Communications Server for Linux configuration servers that the client should contact, in order of preference. Use the same format as for *Server1*.

If the client has tried to contact a server using a UDP broadcast (or has tried to contact the server specified in *Server1*), but has received no response, it then attempts to contact the server specified in *Server2* using a directed message. If this fails, it tries the server specified in *Server3*, and so on. These server names are optional, but provide a backup mechanism if the broadcast method of locating a server fails or if the server specified by *Server1* is unavailable.

If the client tries all the servers listed without success, it waits for the number of seconds specified by the *server_lost_timeout* parameter, then restarts the process of trying to contact a server (either with UDP broadcasts or with the first server listed).

The parameters *Server2–Server9* cannot be set to * to indicate the use of UDP broadcasts. Only the *Server1* parameter can be used to indicate this, because the * value must precede any server names in the file.

Logging

The Logging subkey specifies logging options for the client. These options can be used to specify client logging settings that override the logging options specified for the domain as a whole. For more information about specifying domain logging options, see "Configuring Logging" on page 55.

If central logging is enabled, all log messages are written to a central file on a server. In this case, only the *exception_logging_enabled* and *audit_logging_enabled* parameters specified here are used; the remaining parameters are ignored.

The logging options are specified as follows:

exception_logging_enabled

The Registry data type of this value is REG_SZ.

Set this parameter to one of the following values:

YES Record exception messages.

NO Do not record exception messages.

This parameter is optional. If it is not specified, the client uses the global domain settings to determine whether exception messages are recorded. (The initial default is that exception messages are recorded.)

audit_logging_enabled

The Registry data type of this value is REG_SZ.

Set this parameter to one of the following values:

YES Record audit messages.

NO Do not record audit messages.

This parameter is optional. If it is not specified, the client uses the global domain settings to determine whether audit messages are recorded. (The initial default is that audit messages are recorded.)

log_directory
The Registry data type of this value is REG_SZ.

The full path of the directory where log files are stored on this client. All the log files and backup log files (specified in the following parameters) are stored in this directory. If you are using the log filtering facility described in *Communications Server for Linux Diagnostics Guide*, the file **logfilter.txt** (which controls this facility) is also stored in this directory.

This parameter is optional. If it is not specified, the files are stored in the Windows installation directory.

error_file
The Registry data type of this value is REG_SZ.

Name of the file to which error messages are written. This parameter is optional. If it is not specified, the default is **sna.err**.

To log error and audit messages to a single file, specify the same file name for both this parameter and the *audit_file* parameter.

backup_error_file
The Registry data type of this value is REG_SZ.

Name of the backup error log file. When the error log file reaches the size specified in *error_file_wrap_size*, Communications Server for Linux copies its contents to the backup file (overwriting any existing file), then clears the error log file.

This parameter is optional. If it is not specified, the default is **bak.err**.

To log error and audit messages to a single file, specify the same file name for both this parameter and the *backup_audit_file* parameter.

error_file_wrap_size
The Registry data type of this value is REG_DWORD.

The maximum size of the log file specified by *error_file*. When a message written to the file causes the file size to exceed this limit, Communications Server for Linux copies the current contents of the log file to the backup log file, then clears the log file. This means that the maximum amount of disk space taken up by error log files is approximately twice the value of the *error_file_wrap_size* parameter.

This parameter is optional. If it is not specified, the default is 1000000 (bytes). If you are logging error and audit messages to the same file, this parameter must be set to the same value as the *audit_file_wrap_size* parameter.

audit_file
The Registry data type of this value is REG_SZ.

Name of the file to which audit messages are written. This parameter is optional. If it is not specified, the default is **sna.aud**.

To log error and audit messages to a single file, specify the same file name for both this parameter and the *error_file* parameter.

backup_audit_file
The Registry data type of this value is REG_SZ.

Name of the backup audit log file. When the audit log file reaches the size specified in *audit_file_wrap_size*, Communications Server for Linux copies its contents to the backup file (overwriting any existing file), then clears the audit log file.

This parameter is optional. If it is not specified, the default is **bak.aud**.

To log error and audit messages to a single file, specify the same file name for both this parameter and the *backup_error_file* parameter.

audit_file_wrap_size

The Registry data type of this value is REG_DWORD.

The maximum size of the log file specified by *audit_file*. When a message written to the file causes the file size to exceed this limit, Communications Server for Linux copies the current contents of the log file to the backup log file and clears the log file. This means that the maximum amount of disk space taken up by audit log files is approximately twice the value of the *audit_file_wrap_size* parameter.

This parameter is optional. If it is not specified, the default is 1000000 (bytes). If you are logging error and audit messages to the same file, this parameter must be set to the same value as the *error_file_wrap_size* parameter.

succinct_errors

The Registry data type of this value is REG_SZ.

Specifies whether to use succinct logging or verbose logging in the error log file. This setting applies to both exception logs and problem logs. You can specify either of the following values:

YES Use succinct logging: each message in the log file contains a summary of the message header information (such as the message number and log type) and the message text string and parameters. To obtain more details of the cause of the log and any action required, you can use the **snahelp** utility on a computer running Linux.

NO Use verbose logging: each message in the log file includes a full listing of the message header information, the message text string and parameters, and additional information on the cause of the log and any action required.

This parameter is optional. If it is not specified, the default is taken from the previous **set_global_log_type** command issued to the master server (or set using the Motif administration program). The initial default, before any **set_global_log_type** command has been issued, is to use succinct logging.

If you are using central logging, the choice of succinct or verbose logging for messages from all computers is determined by the setting of this parameter on the server acting as the central logger; this setting may either be from the **set_global_log_type** command, or from a **set_log_type** command issued to that server to override the default.

succinct_audits

The Registry data type of this value is REG_SZ.

Specifies whether to use succinct logging or verbose logging in the audit log file. The permitted values and their meanings are the same as for the *succinct_errors* parameter.

API_tracing

The API_tracing subkey specifies API tracing options for applications running on the client. For more information about tracing, refer to *Communications Server for Linux Diagnostics Guide*. The tracing options are specified as follows:

file1 The Registry data type of this value is REG_SZ.

The full path name of the trace file, or of the first trace file if tracing is to two files (see the description of the *file2* parameter).

This parameter is required if you want to enable API tracing.

file2 The Registry data type of this value is REG_SZ.

The full path name of the second trace file. This parameter is optional; to indicate that tracing is to one file instead of two files, do not include this line.

If both *file1* and *file2* are specified, tracing is to two files. When the first file reaches the size specified by the *flip_size* parameter, the second file is cleared, and tracing continues to the second file. When this file then reaches the size specified by *flip_size*, the first file is cleared, and tracing continues to the first file. This ensures that tracing can continue for long periods without using excessive disk space; the maximum space required is approximately twice the value of the *flip_size* parameter.

flip_size

The Registry data type of this value is REG_DWORD.

The maximum size of the trace file. If two file names are specified, tracing switches between the two files when the current file reaches this size. If only one file name is specified, this parameter is ignored; the file size is not limited.

This parameter is optional. If it is not specified, the default is 1000000 (bytes).

truncation_length

The Registry data type of this value is REG_DWORD.

The maximum length, in bytes, of the information written to the trace file for each message. If a message is longer than this, Communications Server for Linux writes only the start of the message to the trace file, and discards the data beyond *truncation_length*. This enables you to record the most important information for each message but avoid filling up the file with long messages.

This parameter is optional. If it is not specified, Communications Server for Linux does not truncate messages (all the data from each message is written to the file).

all_api The Registry data type of this value is REG_SZ.

To trace messages for all APIs, set this parameter to YES. In this case, Communications Server for Linux ignores the parameters from *appc* through *nof*.

To disable tracing for all APIs, set *all_api* and all of the parameters from *appc* through *nof* to NO.

To trace only messages for specific APIs, set *all_api* to NO, and use the parameters from *appc* through *nof* to indicate which APIs to trace.

This parameter is optional. If it is not specified, the default is NO.

appc The Registry data type of this value is REG_SZ.

To trace APPC API messages, set this parameter to YES; otherwise, set it to NO.

This parameter is optional. If it is not specified, the default is NO. If the *all_api* parameter is set to YES, this parameter is ignored, and APPC messages are traced.

cpic The Registry data type of this value is REG_SZ.

To trace CPI-C API messages, set this parameter to YES; otherwise, set it to NO.

This parameter is optional. If it is not specified, the default is NO. If the *all_api* parameter is set to YES, this parameter is ignored, and CPI-C messages are traced.

csv The Registry data type of this value is REG_SZ.

To trace CSV API messages, set this parameter to YES; otherwise, set it to NO.

This parameter is optional. If it is not specified, the default is NO. If the *all_api* parameter is set to YES, this parameter is ignored, and CSV messages are traced.

rui The Registry data type of this value is REG_SZ.

To trace LUA RUI messages, set this parameter to YES; otherwise, set it to NO.

This parameter is optional. If it is not specified, the default is NO. If the *all_api* parameter is set to YES, this parameter is ignored, and LUA RUI messages are traced.

nof The Registry data type of this value is REG_SZ.

To trace NOF API messages, set this parameter to YES; otherwise, set it to NO. NOF messages are not used directly by applications on Windows clients, but are used internally by Communications Server for Linux components in obtaining configuration information.

This parameter is optional. If it is not specified, the default is NO. If the *all_api* parameter is set to YES, this parameter is ignored, and NOF messages are traced.

CS_tracing

The CS_tracing subkey specifies options for client/server tracing (tracing on messages between the client and Communications Server for Linux servers). For more information about tracing, refer to *Communications Server for Linux Diagnostics Guide*. The tracing options are specified as follows:

file1 The Registry data type of this value is REG_SZ.

The full path name of the trace file, or of the first trace file if tracing is to two files (see the description of the *file2* parameter).

This parameter is required if you want to enable client/server tracing; you also need to set the *trace_flags* parameter.

file2 The Registry data type of this value is REG_SZ.

The full path name of the second trace file. This parameter is optional; to indicate that tracing is to one file instead of two files, do not include this line.

If both *file1* and *file2* are specified, tracing is to two files. When the first file reaches the size specified by the *flip_size* parameter, the second file is cleared, and tracing continues to the second file. When this file then reaches the size specified by *flip_size*, the first file is cleared, and tracing continues to the first file. This ensures that tracing can continue for long periods without using excessive disk space; the maximum space required is approximately twice the value of the *flip_size* parameter.

flip_size

The Registry data type of this value is `REG_DWORD`.

The maximum size of the trace file. If two file names are specified, tracing switches between the two files when the current file reaches this size. If only one file name is specified, this parameter is ignored; the file size is not limited.

This parameter is optional. If it is not specified, the default is `1000000` (bytes).

admin_msg

The Registry data type of this value is `REG_SZ`.

To trace internal messages relating to client/server topology, set this parameter to `YES`; otherwise, set it to `NO`.

This parameter is optional. If it is not specified, the default is `NO`.

datagram

The Registry data type of this value is `REG_SZ`.

To trace datagram messages, set this parameter to `YES`; otherwise, set it to `NO`.

This parameter is optional. If it is not specified, the default is `NO`.

data The Registry data type of this value is `REG_SZ`.

To trace data messages, set this parameter to `YES`; otherwise, set it to `NO`.

This parameter is optional. If it is not specified, the default is `NO`.

send The Registry data type of this value is `REG_SZ`.

To trace all data messages sent from the client to the server, set this parameter to `YES`; otherwise, set it to `NO`.

This parameter is optional. If it is not specified, the default is `NO`.

receive The Registry data type of this value is `REG_SZ`.

To trace all data messages received by the client from the server, set this parameter to `YES`; otherwise, set it to `NO`.

This parameter is optional. If it is not specified, the default is `NO`.

Internal_tracing

The `Internal_tracing` subkey specifies options for tracing the internal operation of the client. For more information about tracing, refer to *Communications Server for Linux Diagnostics Guide*. The tracing options are specified as follows:

file1 The Registry data type of this value is `REG_SZ`.

The full path name of the trace file, or of the first trace file if tracing is to two files (see the description of the *file2* parameter).

This parameter is required if you want to enable internal tracing; you also need to set the *trace_level* parameter.

file2 The Registry data type of this value is REG_SZ.

The full path name of the second trace file. This parameter is optional; to indicate that tracing is to one file instead of two files, do not include this line.

If both *file1* and *file2* are specified, tracing is to two files. When the first file reaches the size specified by the *flip_size* parameter, the second file is cleared, and tracing continues to the second file. When this file then reaches the size specified by *flip_size*, the first file is cleared, and tracing continues to the first file. This ensures that tracing can continue for long periods without using excessive disk space; the maximum space required is approximately twice the value of the *flip_size* parameter.

flip_size

The Registry data type of this value is REG_DWORD.

The maximum size of the trace file. If two file names are specified, tracing switches between the two files when the current file reaches this size. If only one file name is specified, this parameter is ignored; the file size is not limited.

This parameter is optional. If it is not specified, the default is 1000000 (bytes).

trace_level

The Registry data type of this value is REG_DWORD.

The level of detail included in the trace. The range of valid values is from 0 (all tracing) to 20 (no tracing).

This parameter is optional. If it is not specified, the default is 20 (no tracing).

trace_flushing

The Registry data type of this value is REG_SZ.

If this parameter is set to YES, each trace statement is flushed to disk immediately. This slows operation considerably, but ensures that trace data is not lost if a crash occurs.

This parameter is optional. If it is not specified, the default is NO.

Appl_Name

The *Appl_Name* subkey specifies options for a CPI-C application.

Note: These options can be specified either in environment variables or in the registry. Communications Server for Linux checks the environment variable first, and uses this information if it is specified; it uses the registry entry only if the environment variable is not specified. You may need to use environment variables if you are using Windows Terminal Server and need to run multiple copies of the same application using different local LUs.

To set these options in the registry for one or more applications, include a section in this format for each application, and replace the *Appl_Name* variable with the application program's executable name (not including the **.exe** file name extension).

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

The options are specified as follows:

APPCLLU
This option can be specified using the APPCLLU environment variable instead of in the registry.

The Registry data type of this value is REG_SZ.

The name of the local LU that this application uses.

This parameter is optional. If it is not specified, the application attempts to use the default LU (the LU associated with a local node's control point).

APPCTPN
This option can be specified using the APPCTPN environment variable instead of in the registry.

The Registry data type of this value is REG_SZ.

The TP name of the application. This name is used in log and trace files to identify the application. For an invoked application (one that issues Accept_Conversation), it is also used to match the TP name on an incoming Allocate request with the correct application; the invoked application can also use the Specify_Local_TP_Name call to specify additional names to be matched with incoming Allocate requests.

This parameter is optional. If it is not specified, the default is CPIC_DEFAULT_TPNAME.

CSV_data

The CSV_data subkey specifies options for applications that use the CSV interface. It applies only to applications that use the CONVERT verb to perform character conversion with a user-defined conversion table (Table G). For more information about the CONVERT verb, refer to *Communications Server for Linux CSV Programmer's Guide*.

If no applications on the client use this function, you do not need to include this section.

The only option in this section is as follows:

CSVTBLG
The Registry data type of this value is REG_SZ.

The full path name of the file containing the user-defined Table G conversion table. This parameter is required if CSV applications need to perform Table G character conversion (there is no default); otherwise it is optional.

Managing Remote API Clients on AIX or Linux

UNIX

The Remote API Client can run on AIX, Linux, Linux for pSeries, or Linux for System z.

Client information for a Remote API Client on AIX or Linux is stored in the **sna_clnt.net** file, which is created when you install the SNA software on the client. That file must be present before you can enable the client software.

Note: If the client uses HTTPS to access its servers, you will need to modify the **sna_clnt.net** file to specify the names of these servers and the WebSphere server providing HTTPS access to them before you can use the client. See "Client Network Data File (sna_clnt.net)" for more information.

Enabling and disabling Remote API Clients on AIX or Linux

To enable the Remote API Client software on AIX or Linux, enter the following command at the command prompt:

sna start [**-t**]

When you install the client, the installation utility automatically updates the startup file **/etc/rc.sna** (AIX) or **/etc/rc.d/init.d/snastart** (Linux) to include the **sna start** command. This ensures that the client is started automatically at system startup. If you do not want it to be started automatically, you can remove or comment out this line, and then follow the instructions in this section to enable the software manually.

The only option is as follows:

-t Activates client/server tracing. This enables you to diagnose problems that occur during the client's attempt to connect to a server. If you do not use this option, client/server tracing is inactive at all interfaces; you can then activate it as required, using the command-line administration program **snaadmin**.

This option is equivalent to selecting the *Set all tracing on* field in the Motif administration program, except that it does not enable DLC tracing.

Tracing degrades the performance of Communications Server for Linux components. After the software is enabled, you can use the command-line administration program **snaadmin** to stop tracing when it is no longer required. For more information about tracing, refer to *Communications Server for Linux Diagnostics Guide*.

To stop the Remote API Client, enter the following command at the command prompt:

sna stop

Client Network Data File (sna_clnt.net)

The **sna_clnt.net** file defines the Communications Server for Linux facilities available on a Remote API Client on AIX or Linux, and the servers the client can access. (For information about the equivalent file on a Windows client, see Chapter 10, "Managing Communications Server for Linux Client/Server Systems," on page 123.)

It also includes information about setting up the IP port numbers that Communications Server for Linux uses for client/server communications. The default port numbers should be suitable in most cases; you need to refer to this

information only if Communications Server for Linux logs error messages indicating that there is a port number clash with another program on the same computer.

A client computer does not hold a copy of the domain configuration file or the SNA network data file; it holds only the information it needs to access servers on the Communications Server for Linux LAN, and relies on a server to provide the necessary configuration information.

The SNA network information required is held in the file **sna_clnt.net**, which is stored in the directory **/etc/sna** on AIX or **/etc/opt/ibm/sna** on Linux. This file is set up during the client installation process; it is an ASCII text file that can be modified later as required using a standard text editor.

Note: After changing the parameters in this file, you need to stop and restart the client before your changes take effect. For details of how to do this, see "Enabling and disabling Remote API Clients on AIX or Linux" on page 143.

The contents of the file are as follows:

```
domain = domain_name
maximum_process_count = nn
maximum_header_count = nn
maximum_element_count = nn
invoked_tps = YES | NO
lan_access_timeout = nn
broadcast_attempt_count = nn
server_lost_timeout = nn
*
[ webservername : [ portnumber : ] ]servername1
[ webservername : [ portnumber : ] ]servername2
.
.
.
```

The following list describes the parameters in each line of the file:

domain The *domain_name* parameter value indicates the domain name of the Communications Server for Linux LAN; this name is set to `ibmcs_domain`during the client installation. This line is required.

maximum_process_count
Specify the maximum total number of APPC, CPI-C, LUA and NOF applications that can run on this client at any one time.

This parameter is optional; the default value is `240`, which should normally be sufficient. If you see error messages reporting a failure to allocate an IPC control block, you may need to increase the maximum process count by specifying this parameter; the largest value you can specify is `4096`.

maximum_header_count, *maximum_element_count*
These two parameters are optional; the default values are `1250` and `1800`, which should normally be sufficient. You will not normally need to supply values for these parameters except as instructed by support personnel.

invoked_tps
Specify `invoked_tps = YES` if this client is used to run invoked TPs (APPC TPs that issue the RECEIVE_ALLOCATE verb, or CPI-C applications that issue the Accept_Conversation or Accept_Incoming verbs). In this case, you may also need to define the TP on this client. For more information, see "Defining TPs" on page 86.

Specify `invoked_tps = NO` if this client is not used to run invoked TPs.

This line is optional; if it is not included, the default is `NO`.

lan_access_timeout
Specify the time in seconds for which the IP or HTTPS connection from the client to a server should be kept active while no applications on the client are using Communications Server for Linux resources. For more information, see "LAN Access Timeout" on page 127.

The minimum timeout is 60 seconds (lower values are rounded up to 60 seconds). To bring down the connection more quickly, disable the Communications Server for Linux software on the client.

To indicate no timeout, so that the connection is kept active as long as the Communications Server for Linux software is running on the client, do not specify this parameter.

This parameter is optional; if it is not specified, the default is no timeout.

broadcast_attempt_count
If the client uses the broadcast method to contact a server (specified by the * entry), this parameter specifies the maximum number of broadcasts to be made in one attempt to contact a server. The minimum value is 1; if a higher value is specified, the client retries every 10 seconds until it contacts a server or until this count is reached. If the count is reached without contacting a server, the client then attempts to contact a named server.

This parameter is optional; if it is not specified, the default is 5.

server_lost_timeout
If the client loses contact with a server and needs to reconnect, or if it has failed to contact a server using either broadcasts or named servers, this parameter specifies the time in seconds for which the client waits before beginning or restarting the attempt to contact a server. If the client has lost contact with the server, Communications Server for Linux does not wait for the full timeout period, but retries after a random period between 5 seconds and the specified timeout; this is to avoid bursts of network traffic caused by large numbers of clients attempting to contact a server at the same time.

This parameter is optional; if it is not specified, the default is 200 seconds.

*
This line indicates that the client is on the same private network as its servers and should attempt to contact a server running Communications Server for Linux by using a UDP broadcast message to all computers on its TCP/IP subnet (or on all subnets that it can access, if the client computer contains more than one LAN adapter card).

This option is available only if the client uses IPv4 addressing. UDP broadcasts are not supported for IPv6.

The client retries the broadcast every 10 seconds, up to the number of attempts specified by the *broadcast_attempt_count* parameter, until it contacts a server. If the limit specified by *broadcast_attempt_count* is reached before a server has been contacted, the client then tries using directed messages to one or more named servers (specified by the following lines of the file).

The use of UDP broadcasts is optional; to specify that broadcasts should not be attempted, do not include this line. If this line is included, it must precede any server names in the file.

In situations where the client is on the same private network as its servers but cannot reach any servers using UDP broadcasts, do not include this line. This applies in the following cases:

- When the Communications Server for Linux LAN spans multiple TCP/IP subnets, and there are no Communications Server for Linux servers in any TCP/IP subnet that the client can access using UDP
- When UDP support is not installed on the client

If the client uses HTTPS to access its servers, UDP broadcasts are not supported. In this case, specify the server names explicitly as described below.

server names

Specify the names of one or more Communications Server for Linux configuration servers that the client should contact to obtain configuration information. This list should contain the names of the master configuration server and any backup servers in the same domain as the client. For information about configuring master and backup servers, see "Configuring Client/Server Functions" on page 53.

If the client uses IPv6 addressing, UDP broadcasts are not supported. You must specify at least one server name instead of using the * option.

If the client uses HTTPS to access its servers, UDP broadcasts are not supported. In this case, specify the name of the WebSphere server that provides HTTPS support and the name of the Communications Server for Linux server, in the following format:

webservername **:** *servername1*

This assumes that WebSphere is set up to use the default port 443 for HTTPS connections. If your network administrator has configured WebSphere to use a different port number, include the port number in the following format:

webservername **:** *portnumber* **:** *servername1*

For more details about configuring WebSphere to support HTTPS connections, refer to *Communications Server for Linux Quick Beginnings*.

If the * line (to indicate the use of UDP broadcasts) is not included, or if the client tried to contact a server using this method but received no response, the client attempts to contact the first server listed using a directed message. If this fails, the client tries the second server listed, and so on. This means that you can balance the load between two or more configuration servers by changing the order in which the servers are listed.

If the * line (to indicate the use of UDP broadcasts) is not included, at least one server name must be specified; otherwise, server names are optional.

If the client tries all the servers listed without success, it waits for the time specified by *server_lost_timeout* above, and then restarts the process of trying to contact a server (either with UDP broadcasts or with the first server listed).

In addition to the **sna_clnt.net**, an additional file **server.current** is stored in the same directory (**/etc/sna** on AIX or **/etc/opt/ibm/sna** on Linux). This is a text file containing the name of the server, if any, to which the client is currently connected. You can check this file to determine which server is acting as the client's connection point into the domain.

Defining Client TPs

For information about defining TPs on a Remote API Client system, see "Defining TPs" on page 86 or Appendix B, "Configuring an Invokable TP from the Command Line," on page 171.

Appendix A. Configuration Planning Worksheets

This appendix provides worksheets for configuring specific functions of Communications Server for Linux. The worksheets summarize the basic configuration parameters needed to enable each function; for information about advanced configuration parameters, see the appropriate section in the body of this book, or refer to *Communications Server for Linux Administration Command Reference*.

To gather all of the information needed to configure a node, you must complete worksheets in the following categories:

Node configuration
Complete one of the worksheets contained in "Node Worksheets," depending on the capabilities of the node and the characteristics of the network in which it operates.

Connectivity configuration
Complete one or more of the worksheets contained in "Connectivity Worksheets" on page 151, depending on the link protocols used to communicate with the other systems in your network.

Passthrough services configuration
Complete the worksheets in "Passthrough Services Worksheets" on page 161, for any passthrough services to be supported by the node.

Application support configuration
Complete one or more of the worksheets contained in "User Application Support Worksheets" on page 165, depending on the types of user applications to be supported by the node.

Node Worksheets

Complete only one of the following worksheets:

- "APPN Network Node"
- "APPN End Node" on page 150
- "APPN Branch Network Node" on page 150
- "LEN Node" on page 151

APPN Network Node

Complete this worksheet if the local node is an APPN network node (a node that provides routing services in an APPN network).

Motif Field	Valid Entry/Notes	Your Implementation Value
Node Parameters Dialog		
APPN support	`Network node`	
Control point name	*NETNAME*.*CPNAME* (each 1–8 type A EBCDIC characters) To connect to a VTAM host, this name must match the *NETID*= and *CPNAME*= entries in the VTAM PU statement.	

Motif Field	Valid Entry/Notes	Your Implementation Value
Control point alias	Up to 8 characters	
Node ID	8 hexadecimal digits	
Connectivity Configuration: See "Connectivity Worksheets" on page 151.		
Client/Server Configuration: Not required for a standalone node.		
Configuration server?	Should the node act as a configuration server, to store information about domain resources in the Communications Server for Linux LAN?	
Application Configuration: See "User Application Support Worksheets" on page 165.		

APPN End Node

Complete this worksheet if the local node is an APPN end node (a node that can use dynamic routing information but does not provide routing services for other nodes).

Motif Field	Valid Entry/Notes	Your Implementation Value
Node Parameters Dialog		
APPN support	`End node`	
Control point name	*NETNAME*.*CPNAME* (each 1–8 type A EBCDIC characters) To connect to a VTAM host, this name must match the *NETID*= and *CPNAME*= entries in the VTAM PU statement.	
Control point alias	Up to 8 characters	
Node ID	8 hexadecimal digits	
Connectivity Configuration: See "Connectivity Worksheets" on page 151.		
Client/Server Configuration: Not required for a standalone node.		
Configuration server?	Should the node act as a configuration server, to store information about domain resources in the Communications Server for Linux LAN?	
Application Configuration: See "User Application Support Worksheets" on page 165.		

APPN Branch Network Node

Complete this worksheet if the local node is an APPN branch network node (a node that provides network node functions to end nodes in a branch separated from the main APPN network, while acting as an end node in the main network itself).

Motif Field	Valid Entry/Notes	Your Implementation Value
Node Parameters Dialog		
APPN support	`Branch network node`	

Motif Field	Valid Entry/Notes	Your Implementation Value
Control point name	*NETNAME*.*CPNAME* (each 1–8 type A EBCDIC characters) To connect to a VTAM host, this name must match the *NETID*= and *CPNAME*= entries in the VTAM PU statement.	
Control point alias	Up to 8 characters	
Node ID	8 hexadecimal digits	
Connectivity Configuration: See "Connectivity Worksheets."		
Client/Server Configuration: Not required for a standalone node.		
Configuration server?	Should the node act as a configuration server, to store information about domain resources in the Communications Server for Linux LAN?	
Application Configuration: See "User Application Support Worksheets" on page 165.		

LEN Node

Complete this worksheet if the local node is a LEN node (a node that does not support APPN functions or a standalone system that communicates only with a host computer).

Motif Field	Valid Entry/Notes	Your Implementation Value
Node Parameters Dialog		
APPN support	`LEN node`	
Control point name	*NETNAME*.*CPNAME* (each 1–8 type A EBCDIC characters) To connect to a VTAM host, this name must match the *NETID*= and *CPNAME*= entries in the VTAM PU statement.	
Control point alias	Up to 8 characters	
Node ID	8 hexadecimal digits	
Connectivity Configuration: See "Connectivity Worksheets."		
Client/Server Configuration: Not required for a standalone node.		
Configuration server?	Should the node act as a configuration server, to store information about domain resources in the Communications Server for Linux LAN?	
Application Configuration: See "User Application Support Worksheets" on page 165.		

Connectivity Worksheets

For each link protocol used to communicate with another node, complete one of the following worksheets. If necessary, you can configure more than one link station on a port.

- "SDLC" on page 152
- "Token Ring" on page 154

SDLC

Complete this worksheet to support connectivity using the SDLC link protocol.

Motif Field	Valid Entry/Notes	Your Implementation Value
SDLC Port Dialog		
SNA port name	Up to 8 characters	
SDLC card number	`0` to *number_of_cards_minus_1*	
Port number	`0` to *number_of_ports_on_card_minus_1*	
Initially active	Select if needed	
Line Details		

Type	`Leased line`
	`Switched outgoing`
	`Switched incoming`	

Link role	`Negotiable`
	`Primary`
	`Primary multi-drop`
	`Secondary`
	`Secondary multi-PU`	

For switched incoming or leased line:		
Poll address	Only for nonprimary, switched incoming ports On a VTAM host, the poll address is configured as the *ADDR*= parameter in the VTAM PU definition. On an AS/400 system, the poll address is the *STNADR* parameter of the Line Description.	
SDLC Link Station Dialog		
Link station fields		
Name	Up to 8 characters	
SNA port name	Up to 8 characters	

Activation	`By administrator`
	`On node startup`
	`On demand`	

Motif Field	Valid Entry/Notes	Your Implementation Value

LU traffic	`Any`
	
	`Independent only`
	
	`Dependent only`	

Independent LU traffic		
Remote node	*NETNAME*.*CPNAME* (each 1–8 type A EBCDIC characters; optional) If the remote system is a VTAM host, you can find the network name (the first eight characters of the fully qualified name) in the *NETID* parameter of the VTAM start list. The last eight characters are in the *SSCPNAME* parameter of the VTAM start list.	

Remote node type	`Discover`
	
	`Network node`
	
	`End or LEN node`	

Dependent LU traffic		

Remote node role	`Host`
	
	`Downstream (SNA Gateway)`
	
	`Downstream (DLUR)`	

Local node ID	8 hexadecimal digits (defaults to node name) In a VTAM configuration, the first three digits should match the *IDBLK* parameter in the PU definition, and the last five should match the *IDNUM* parameter. On an AS/400 system, the node ID is configured in the *EXCHID* parameter.	
Remote node ID	8 hexadecimal digits (optional)	
Downstream PU name	1–8 type A EBCDIC characters	
Upstream DLUS name	*NETNAME*.*LUNAME* (each 1–8 type A EBCDIC characters)	
Contact information		

Motif Field	Valid Entry/Notes	Your Implementation Value
Poll address	For switched incoming ports, only configured on the port. 2 hexadecimal digits: • `C1` for point-to-point • `0xFF` for primary switched outgoing (destination address unknown) • Unique addresses for primary to multi-drop On a VTAM host, the poll address is configured as the *ADDR*= parameter in the VTAM PU definition. On an AS/400 system, the poll address is the *STNADR* parameter of the Line Description.	

Token Ring

Complete this worksheet to support connectivity using the token ring link protocol.

Motif Field	Valid Entry/Notes	Your Implementation Value
Token Ring SAP Dialog		
SNA port name	Up to 8 characters	
Token ring card number	`0` to *number_of_cards_minus_1*	
Local SAP number	Hexadecimal (multiple of 4)	
Initially active	Select if needed	
Define on connection network	Select if needed	
CN name	*NETNAME*.*CNNAME* (each 1–8 type A EBCDIC characters)	
Token Ring Link Station Dialog		
Link station fields		
Name	Up to 8 characters	
SNA port name	Up to 8 characters	

Activation	`By administrator`
	
	`On node startup`
	
	`On demand`	

LU traffic	`Any`
	
	`Independent only`
	
	`Dependent only`	

Independent LU traffic		

Motif Field	Valid Entry/Notes	Your Implementation Value
Remote node	*NETNAME*.*CPNAME* (each 1–8 type A EBCDIC characters; optional) If the remote system is a VTAM host, you can find the network name (the first eight characters of the fully qualified name) in the *NETID* parameter of the VTAM start list. The last eight characters are in the *SSCPNAME* parameter of the VTAM start list.	

Remote node type	`Discover`
	
	`End or LEN node`
	
	`Network node`	

Dependent LU traffic		

Remote node role	`Host`
	
	`Downstream (SNA Gateway)`
	
	`Downstream (DLUR)`	

Local node ID	8 hexadecimal digits (defaults to node name) In a VTAM configuration, the first three digits should match the *IDBLK* parameter in the PU definition, and the last five should match the *IDNUM* parameter. On an AS/400 system, the node ID is configured in the *EXCHID* parameter.	
Remote node ID	8 hexadecimal digits (optional)	
Downstream PU name	1–8 type A EBCDIC characters	
Upstream DLUS name	*NETNAME*.*LUNAME* (each 1–8 type A EBCDIC characters)	
Contact information		
MAC address	Hexadecimal digits If the remote end of this link is a VTAM host, you can find its MAC address in the *MACADDR*= parameter of the VTAM Port definition. If you are configuring a link to an AS/400 system, the MAC address is the *ADPTADR* parameter in the Line Description.	

Motif Field	Valid Entry/Notes	Your Implementation Value
SAP number	Hexadecimal (multiple of 4) If the remote end of this link is a VTAM host, the SAP number is the *SAPADDR*= parameter of the VTAM PU definition. If you are configuring a link to an AS/400 system, the SAP number is the *SSAP* parameter in the Line Description.	

Ethernet

Complete this worksheet to support connectivity using the Ethernet link protocol.

Motif Field	Valid Entry/Notes	Your Implementation Value
Ethernet SAP Dialog		
SNA port name	Up to 8 characters	
Ethernet card number	`0` to *number_of_cards_minus_1*	
Local SAP number	Hexadecimal (multiple of 4)	
Initially active	Select if needed	
Define on connection network	Select if needed	
CN name	*NETNAME*.*CNNAME* (each 1–8 type A EBCDIC characters)	
Ethernet type	Select Standard or 802.3	
Ethernet Link Station Dialog		
Link station fields		
Name	Up to 8 characters	
SNA port name	Up to 8 characters	

Activation	`By administrator`
	
	`On node startup`
	
	`On demand`	

LU traffic	`Any`
	
	`Independent only`
	
	`Dependent only`	

Independent LU traffic		

Motif Field	Valid Entry/Notes	Your Implementation Value
Remote node	*NETNAME*.*CPNAME* (each 1–8 type A EBCDIC characters; optional) If the remote system is a VTAM host, you can find the network name (the first eight characters of the fully qualified name) in the *NETID* parameter of the VTAM start list. The last eight characters are in the *SSCPNAME* parameter of the VTAM start list.	

Remote node type	`Discover`
	
	`Network node`
	
	`End or LEN node`	

Dependent LU traffic		

Remote node role	`Host`
	
	`Downstream (SNA Gateway)`
	
	`Downstream (DLUR)`	

Local node ID	8 hexadecimal digits (defaults to node name) In a VTAM configuration, the first three digits should match the *IDBLK* parameter in the PU definition, and the last five should match the *IDNUM* parameter. On an AS/400 system, the node ID is configured in the *EXCHID* parameter.	
Remote node ID	8 hexadecimal digits (optional)	
Downstream PU name	1–8 type A EBCDIC characters	
Upstream DLUS name	*NETNAME*.*LUNAME* (each 1–8 type A EBCDIC characters)	
Contact information		
MAC address	Hexadecimal digits If the remote end of this link is a VTAM host, you can find its MAC address in the *MACADDR*= parameter of the VTAM Port definition. If you are configuring a link to an AS/400 system, the MAC address is the *ADPTADR* parameter in the Line Description.	

Motif Field	Valid Entry/Notes	Your Implementation Value
SAP number	Hexadecimal (multiple of 4) If the remote end of this link is a VTAM host, the SAP number is the *SAPADDR*= parameter of the VTAM PU definition. If you are configuring a link to an AS/400 system, the SAP number is the *SSAP* parameter in the Line Description.	

QLLC (X.25)

Complete this worksheet to support connectivity using the QLLC (X.25) link protocol.

Motif Field	Valid Entry/Notes	Your Implementation Value
QLLC Port Dialog		
SNA port name	Up to 8 characters	
X.25 card number	`0` to *number_of_cards_minus_1*	
Initially active	Select if needed	
QLLC Link Station Dialog		
Link station fields		
Name	Up to 8 characters	
SNA port name	Up to 8 characters	

Activation	`By administrator`
	
	`On node startup`
	
	`On demand`	

LU traffic	`Any`
	
	`Independent only`
	
	`Dependent only`	

Independent LU traffic		
Remote node	*NETNAME*.*CPNAME* (each 1–8 type A EBCDIC characters; optional) If the remote system is a VTAM host, you can find the network name (the first eight characters of the fully qualified name) in the *NETID* parameter of the VTAM start list. The last eight characters are in the *SSCPNAME* parameter of the VTAM start list.	

Motif Field	Valid Entry/Notes	Your Implementation Value

Remote node type	`Discover`
	`Network node`
	`End or LEN node`	

Dependent LU traffic		

Remote node role	`Host`
	`Downstream (SNA Gateway)`
	`Downstream (DLUR)`	

Local node ID	8 hexadecimal digits (defaults to node name) In a VTAM configuration, the first three digits should match the *IDBLK* parameter in the PU definition, and the last five should match the *IDNUM* parameter. On an AS/400 system, the node ID is configured in the *EXCHID* parameter.	
Remote node ID	8 hexadecimal digits (optional)	
Downstream PU name	1–8 type A EBCDIC characters	
Upstream DLUS name	*NETNAME*.*LUNAME* (each 1–8 type A EBCDIC characters)	
Contact information		
Remote X.25 address	Hexadecimal digits (only for SVC); `1–4096` (only for PVC)	

Multipath Channel

Complete this worksheet to support connectivity using the Multipath Channel link protocol.

Motif Field	Valid Entry/Notes	Your Implementation Value
Multipath Channel Port Dialog		
SNA port name	Up to 8 characters	
Port number	Must match the device number of the MultiPath Channel device	
Initially active	Select if needed	
Multipath Channel Link Station Dialog		
Link station fields		
Name	Up to 8 characters	
SNA port name	Up to 8 characters	

Activation	`By administrator`
	`On node startup`
	`On demand`	

Motif Field	Valid Entry/Notes	Your Implementation Value
Independent LU traffic		
Remote node	*NETNAME*.*CPNAME* (each 1–8 type A EBCDIC characters; optional) If the remote system is a VTAM host, you can find the network name (the first eight characters of the fully qualified name) in the *NETID* parameter of the VTAM start list. The last eight characters are in the *SSCPNAME* parameter of the VTAM start list.	

Remote node type	`Discover`
	
	`Network node`
	
	`End or LEN node`	

Enterprise Extender (HPR/IP)

Complete this worksheet to support connectivity using the Enterprise Extender link protocol.

Motif Field	Valid Entry/Notes	Your Implementation Value
Enterprise Extender Port Dialog		
SNA port name	Up to 8 characters	
Initially active	Select if needed	
Protocol	Whether link stations on this port use IPv4 or IPv6 addresses.	
Local IP interface	The identifier for the local network adapter card to be used for the IP link, if you have access to multiple IP networks. If you have access to only one IP network, you can leave this field blank.	
Define on connection network	Select if needed	
CN name	*NETNAME*.*CNNAME* (each 1–8 type A EBCDIC characters)	
Enterprise Extender Link Station Dialog		
Link station fields		
Name	Up to 8 characters	
SNA port name	Up to 8 characters	

Activation	`By administrator`
	
	`On node startup`
	
	`On demand`	

Independent LU traffic		

Motif Field	Valid Entry/Notes	Your Implementation Value
Remote node	*NETNAME*.*CPNAME* (each 1–8 type A EBCDIC characters; optional) If the remote system is a VTAM host, you can find the network name (the first eight characters of the fully qualified name) in the *NETID* parameter of the VTAM start list. The last eight characters are in the *SSCPNAME* parameter of the VTAM start list.	

Remote node type	`Discover`
	
	`End or LEN node`
	
	`Network node`	

Contact information		
Remote IP host name	IPv4 dotted-decimal address (such as `193.1.11.100`), IPv6 colon-hexadecimal address (such as `2001:0db8:0000:0000:0000:0000:1428:57ab` or `2001:db8::1428:57ab`), name (such as `newbox.this.co.uk`), or alias (such as `newbox`). The *protocol* parameter on the port determines whether this is an IPv4 or IPv6 address. If you specify a name or alias, the Linux system must be able to resolve this to a fully qualified name (either using the local TCP/IP configuration or using a Domain Name server).	

Passthrough Services Worksheets

Complete worksheets for any of the passthrough services described in the following sections, if the service is to be supported by the local node:

- "DLUR on the Local Node"
- "Passthrough DLUR for Downstream Nodes" on page 162
- "SNA Gateway" on page 162
- "TN Server" on page 163
- "TN Redirector" on page 164

DLUR on the Local Node

Complete this worksheet to support DLUR on the local node.

Motif Field	Valid Entry/Notes	Your Implementation Value
Node Configuration: See "Node Worksheets" on page 149.		
Connectivity Configuration: See "Connectivity Worksheets" on page 151. To support DLUR on the local node, configure connectivity to the APPN network.		
DLUR PU: .		

Motif Field	Valid Entry/Notes	Your Implementation Value
PU name	1–8 type A EBCDIC characters	
DLUS name	*NETNAME*.*LUNAME* (each 1–8 type A EBCDIC characters)	
Backup DLUS name	This parameter is optional. *NETNAME*.*LUNAME* (each 1–8 type A EBCDIC characters)	
PU ID	8 hexadecimal digits In a VTAM configuration, the first three digits should match the *IDBLK* parameter in the PU definition, and the last five digits should match the *IDNUM* setting. On an AS/400 system, the PU ID is configured in the *EXCHID* parameter.	
Initially active	Select if needed	
Compression supported	Select if needed	
Retry contacting DLUS indefinitely	Select if needed	
Local LU and Application Configuration: See "User Application Support Worksheets" on page 165. You must configure local dependent LUs and any application support you require.		

Passthrough DLUR for Downstream Nodes

If the local node is an APPN network node, you can provide passthrough DLUR services for downstream nodes. Complete this worksheet to support DLUR.

Motif Field	Valid Entry/Notes	Your Implementation Value
Node Configuration: Configure the node as a network node (see "APPN Network Node" on page 149).		
Connectivity Configuration: See "Connectivity Worksheets" on page 151. Configure connectivity to the APPN network and also connectivity for dependent traffic to the downstream nodes.		

SNA Gateway

Complete this worksheet if the local node is to support SNA gateway.

Motif Field	Valid Entry/Notes	Your Implementation Value
Node Configuration: See "Node Worksheets" on page 149.		
Connectivity Configuration: See "Connectivity Worksheets" on page 151. Configure connectivity for dependent traffic to host, and links for dependent traffic to each downstream node.		
Local LU and Application Configuration: See "User Application Support Worksheets" on page 165.		
LU Pool Dialog		

Motif Field	Valid Entry/Notes	Your Implementation Value
Pool name	1–8 type AE EBCDIC characters	
LU lists	Names of the LUs (type 0–3) to assign to the pool	
Downstream LU Dialog		
Downstream LU name	1–8 type A EBCDIC characters (1–5 for the base name for a range of LUs)	
Downstream PU name	type A EBCDIC string	
LU numbers	1–255 (for a range, supply first and last numbers)	
Upstream LU name	Type A EBCDIC string (for LU name) or type AE EBCDIC string (for LU pool name)	

TN Server

Complete this worksheet if the local node is to support TN3270 clients.

Motif Field	Valid Entry/Notes	Your Implementation Value
Node Configuration: See "Node Worksheets" on page 149.		
Connectivity Configuration: See "Connectivity Worksheets" on page 151 (configure for dependent LU traffic to host).		
Local LU and Application Configuration: See "User Application Support Worksheets" on page 165.		
LU Pool Dialog		
Pool name	1–8 type AE EBCDIC characters	
LU lists	Names of the LUs (type 0–3) to assign to the pool	
TN Server Access Dialog		
TN3270 client address	Specify one of the following: • `Default record` (any TN3270 client) • `TCP/IP address` (IP address of client, either IPv4 or IPv6 address) • `TCP/IP name or alias`	
Support TN3270E	Select to support TN3270E (in addition to TN3270 and TN3287)	
TN3270 port and LUs		
TCP/IP port number	Usually 23.	
Display LU Assigned	LU or pool name	
Printer LU Assigned	LU or pool name	
Allow access to specific LU	Select if needed	
SSL secure session	Select if needed	
Perform client authentication	Select if needed	

Motif Field	Valid Entry/Notes	Your Implementation Value
Encryption strength	Specify one of the following: • `Authenticate Only` • `Authenticate Minimum` • `40 Bit Minimum` • `56 Bit Minimum` • `128 Bit Minimum` • `168 Bit Minimum`	
TN Server Association Dialog		
Display LU	LU name	
Printer LU	LU name	

TN Redirector

Complete this worksheet if the local node is to support Telnet clients using TN Redirector.

Motif Field	Valid Entry/Notes	Your Implementation Value
Node Configuration: See "Node Worksheets" on page 149.		
TN Redirector Access Dialog		
Telnet client address	Specify one of the following: • `Default record` (any Telnet client) • `TCP/IP address` (IP address of client, either IPv4 or IPv6 address) • `TCP/IP name or alias`	
TCP/IP port number	Usually `23`.	
SSL secure session	Select if needed	
Perform client authentication	Select if needed	
Encryption strength	Specify one of the following: • `Authenticate Only` • `Authenticate Minimum` • `40 Bit Minimum` • `56 Bit Minimum` • `128 Bit Minimum` • `168 Bit Minimum`	
Host address	Specify one of the following: • `TCP/IP address` (IP address of host, eithe IPv4 or IPv6 address) • `TCP/IP name or alias`	
TCP/IP port number		
SSL secure session	Select if needed	

Motif Field	Valid Entry/Notes	Your Implementation Value
Encryption strength	Specify one of the following: • `Authenticate Only` • `Authenticate Minimum` • `40 Bit Minimum` • `56 Bit Minimum` • `128 Bit Minimum` • `168 Bit Minimum`	

User Application Support Worksheets

Complete the following worksheets if the corresponding user-level applications are to be supported by the local node:

- "APPC"
- "CPI-C" on page 168
- "5250" on page 169
- "3270" on page 169
- "LUA" on page 170

APPC

Complete this worksheet if the local node is to support APPC applications.

Motif Field	Valid Entry/Notes	Your Implementation Value
Node Configuration: See "Node Worksheets" on page 149.		
Connectivity Configuration: See "Connectivity Worksheets" on page 151.		
Local LU Dialog: Not required if you can use the default control point LU.		
LU name *LU alias*	1–8 type A EBCDIC characters Up to 8 characters	
Dependent LU parameters		
Host LS/DLUR PU	Name of dependent link station to host or DLUR PU (must be defined before defining an LU)	
LU number	`1–255` This value must match the *LOCADDR* parameter in the VTAM/NCP LU resource definition statement.	
Member of default pool	Select if needed (only for dependent LU)	
Local LU parameters		
Support syncpoint	Select if needed	
Disable password substitution	Select if needed	

Motif Field	Valid Entry/Notes	Your Implementation Value
Restrict to specific SSCP	Select if needed (only for dependent LU). The SSCP ID is defined in the *SSCPID=* field of the VTAM start list.	
Remote Node Dialog: Only configure if the local node is a LEN node.		
Node's SNA network name	*NETNAME*.*CPNAME* (each 1–8 type A EBCDIC characters)	
Partner LU Dialog: Only required for communication with a LEN node, to define a partner LU alias, or if the local node is a LEN node.		
Partner LU name	(*NETNAME*.*LUNAME* (each 1–8 type A EBCDIC characters)	
Alias	Up to 8 characters	
Uninterpreted name	1–8 type AE EBCDIC characters (if host LU name is different from PLU name used locally)	
Supports parallel sessions	Select if supported	
Location	*NETNAME*.*CPNAME* (each 1–8 type A EBCDIC characters)	
LS Routing Dialog: Only required if partner LU is located by link station.		
LU name	1–8 type A EBCDIC characters	
LS name	Up to 8 characters	
Partner LU name	(*NETNAME*.*LUNAME* (each 1–8 type A EBCDIC characters)	
Use partner LU name as a wildcard	Select if needed	
Mode Dialog: Only required if you are using a nonstandard mode.		
Name	1–8 type A EBCDIC characters	
COS name	1–8 type A EBCDIC characters	
Session limits		
Initial session limit	Up to maximum session limit; recommended value is `8`	
Maximum session limit	Up to `32767`	
Minimum contention winner sessions	Up to maximum session limit; recommended value is `0`.	
Minimum contention loser sessions	Recommended value is `0`.	
Auto-activated sessions	0 to *minimum_contention_winners*	
Receive pacing window		
Initial window size	Recommended value is `4`	
Maximum window size	Optional	
Session timeout		
Maximum RU size	Recommended upper limit is `1024`.	
Compression supported		

Motif Field	Valid Entry/Notes	Your Implementation Value
Max inbound compression	None RLE LZ9 LZ10	
Max outbound compression	None RLE LZ9 LZ10	
Session Security Dialog: Only required if session security is required for sessions between a specific local and partner LU.		
Local LU	1–8 type A EBCDIC characters	
Partner LU	1–8 type A EBCDIC characters	
Password	16-digit hexadecimal number	
TP Invocation Dialog: Only required if local TP is to be started in response to requests from remote systems.		
TP name	User application: up to 64 ASCII characters Service TP: up to 8 hexadecimal digits	
Restrict to specific LU	Select if needed	
LU alias	Up to 8 characters	
Multiple instances supported	Select for nonqueued TPs; if not selected, incoming Allocate requests are queued if the TP is already running	
Route incoming Allocates to running TP	Select for a broadcast queued TP	
Full path to TP executable	Path and file name of the executable file (defaults to *TP name*)	
Arguments	Any valid arguments to the executable	
User ID	Up to 64 characters	
Group ID	Up to 64 characters	
TP Definition Dialog: Defines APPC characteristics.		
TP name	User application: up to 64 ASCII characters Service TP: up to 8 hexadecimal digits	
Conversation level security required	Select to require a valid user name and password on allocation requests	
Restrict access	Select to require that user names be included on a security access list	
Security access list	Name of security access list	
Conversation type	Basic Mapped Either	

Motif Field	Valid Entry/Notes	Your Implementation Value

Sync level	`None`
	`Confirm`
	`Sync-point`
	`None or Confirm`
	`None, Confirm, or Sync-point`	

PIP allowed	Select if needed	
Conversation Security Dialog: Only required if conversation security is required for a local TP that is to be started in response to requests from remote systems.		
User ID	Up to 10 characters	
Password	Up to 10 characters	

CPI-C

Complete this worksheet if the local node is to support CPI-C applications.

Motif Field	Valid Entry/Notes	Your Implementation Value
Node Configuration: See "Node Worksheets" on page 149.		
Connectivity Configuration: See "Connectivity Worksheets" on page 151.		
APPC Configuration: See "APPC" on page 165.		
CPI-C Destination Dialog		
Symbolic destination name	1–8 characters	
Local LU	Alias (up to 8 characters) or fully qualified name (*NETNAME*.*LUNAME*, each 1–8 type A EBCDIC characters)	
Partner LU	Alias (up to 8 characters) or fully qualified name (*NETNAME*.*LUNAME*, each 1–8 type A EBCDIC characters)	
Mode	Type A EBCDIC string	
Partner TP name	User application: up to 64 characters Service TP: up to 8 hexadecimal digits	

Security	`None`
	`Same`
	`Program`	

User ID	Only for security level of `Same` or `Program` (not related to user login ID)	
Password	Only for security level of `Program` (not related to user login password)	

5250

Complete this worksheet if the local node is to support 5250 communications.

Motif Field	Valid Entry/Notes	Your Implementation Value
Node Configuration: See "Node Worksheets" on page 149.		
Connectivity Configuration: See "Connectivity Worksheets" on page 151 (configure for independent traffic).		
APPC Configuration: See "APPC" on page 165.		

3270

Complete this worksheet if the local node is to support 3270 communications.

Motif Field	Valid Entry/Notes	Your Implementation Value
Node Configuration: See "Node Worksheets" on page 149.		
Connectivity Configuration: See "Connectivity Worksheets" on page 151 (configure for dependent traffic).		
LU Type 0–3 Dialog		
LU name	1–8 type A EBCDIC characters (or 1–5 characters for a base name for a range of LUs)	
Host LS/DLUR PU	Name of dependent link station to host or DLUR PU (must be defined before defining an LU)	
LU numbers	1–255 (for a range, supply first and last numbers) This value must match the *LOCADDR* parameter in the VTAM/NCP LU resource definition statement.	

LU type	`3270 model 2 (80x24)` display
	
	`3270 model 3 (80x32)` display
	
	`3270 model 4 (80x43)` display
	
	`3270 model 5 (132x27)` display
	
	`3270 Printer`
	
	`SCS Printer`	

LU in pool	Select desired option (only for display and unrestricted LUs).	
Pool name	1–8 type AE EBCDIC characters	
LU Pool Dialog		
Pool name	1–8 type AE EBCDIC characters	
LU lists	Names of the LUs (type 0–3) to assign to the pool	

LUA

Complete this worksheet if the local node is to support LUA applications.

Motif Field	Valid Entry/Notes	Your Implementation Value
Node Configuration: See "Node Worksheets" on page 149.		
Connectivity Configuration: See "Connectivity Worksheets" on page 151 (configure for dependent traffic).		
LU Type 0–3 Dialog		
LU name	1–8 type A EBCDIC characters (or 1–5 characters for a base name for a range of LUs)	
Host LS/DLUR PU	Name of dependent link station to host or DLUR PU (must be defined before defining an LU)	
LU numbers	1–255 (for a range, supply first and last numbers) This value must match the *LOCADDR* parameter in the VTAM/NCP LU resource definition statement.	
LU type	`Unrestricted`	
LU in pool	Select desired option (only for display and unrestricted LUs)	
Pool name	1–8 type AE EBCDIC characters	
LU Pool Dialog		
Pool name	1–8 type AE EBCDIC characters	
LU lists	Names of the LUs (type 0–3) to assign to the pool	

Appendix B. Configuring an Invokable TP from the Command Line

Communications Server for Linux includes a command-line utility that enables a user or the writer of a TP installation program to define an invokable TP. You can run this utility on a server or client.

The syntax of the command is different depending on whether you are defining, removing, or querying TP definitions, and is different for a Remote API Client on Windows:

Define an invokable TP:

UNIX

snatpinstall -a *file_name*

WINDOWS

tpinst32 -a *file_name*

Note: The **tpinst32** command applies to both 32–bit and x64 versions of Windows.

This command adds one or more TP definitions from the specified *file_name*. If the TP named in the file has already been defined, the information in the file replaces the existing definition. For information about the required file format, see "File Format for an Invokable TP Definition" on page 172.

Remove an invokable TP definition:

UNIX

snatpinstall -r -t *TP_name* [**-l** *LU_alias*]

This command removes the entry that has the specified TP name and (if more than one APPC TP is defined with the same TP name) the specified LU alias. Omit the option **-l** *LU_alias* if the entry is for a CPI-C application, or if there is only one APPC TP defined with the specified TP name.

WINDOWS

tpinst32 -r -t *TP_name*

This command removes the entry that has the specified TP name.

Query invokable TP definitions:

UNIX

snatpinstall -q [**-t** *TP_name*] [**-l** *LU_alias*]

This command queries the entry that has the specified TP name and (if more than one APPC TP is defined with the same TP name) the specified LU alias. Omit the option **-l** *LU_alias* if the entry is for a CPI-C application, or if there is only one APPC TP defined with the specified TP name. If you do not include the option **-t** *TP_name*, the command queries all invokable TP definitions.

WINDOWS

tpinst32 -q -t *TP_name*

This command queries the entry that has the specified TP name. If you do not include the option **-t** *TP_name*, the command queries all invokable TP definitions.

File Format for an Invokable TP Definition

The file that supplies configuration information for an invokable TP is an ASCII text file that can be modified using any standard text editor. Each entry in the file has the following format:

```
[TPname]
PATH          =  full_pathname_of_executable_file
ARGUMENTS     =  command-line_arguments_separated_by_spaces
TYPE          =  QUEUED | QUEUED-BROADCAST | NON-QUEUED
TIMEOUT       =  nnn
```

UNIX

```
USERID        =  user_ID
GROUP         =  group_ID
LUALIAS       =  LU_alias
ENV           =  environment_variable=value
                    .
                    .
                    .
ENV           =  environment_variable=value
```

WINDOWS

```
SHOW          =  MAXIMIZED | MINIMIZED | HIDDEN | NORMAL | NOACTIVATE | MINNOACTIVATE
SECURITY_TYPE =  APPLICATION | SERVICE
SERVICE_NAME  =  name_of_installed_service
USERID        =  domain_name\user_ID
```

The parameters are as follows. For an operator-started TP, the only parameters used are the TP name, the TP type, the timeout value, and (for an APPC TP on AIX or Linux) the LU alias; the other parameters apply only to automatically started TPs.

UNIX

On AIX or Linux, Communications Server for Linux returns an error message if you enter an invalid parameter.

WINDOWS

On Windows machines, Communications Server for Linux ignores invalid parameters.

TPname
The name of the TP (1–64 characters, with no embedded space characters). The TP name specified on the Receive_Allocate, or on the incoming Allocate request, is matched against this name. If the TP is an automatically started TP, it must specify this TP name on the Receive_Allocate when it starts up, to enable Communications Server for Linux to route the incoming Attach to the correct TP.

This name must be enclosed within square brackets. The name can be specified as an ASCII string, enclosed in double quotation marks (for example, `["TPNAME1"]`). Alternatively, it can be specified as a hexadecimal array representing the EBCDIC characters of the TP name (for example, `[<53504E414D45F1>]`) or as a combination of the two (for example, `[<3f>"TP1"]`). In this example, the first character is the unprintable character `0x3f`, and the following characters are `"TP1"`.

Communications Server for Linux converts a supplied ASCII string to EBCDIC, but does not perform any conversion on a hexadecimal string (which is assumed to be in EBCDIC already). It then pads the name with EBCDIC spaces on the right (to a total of 64 characters) before matching against the specified TP name.

PATH The path and file name of the executable file for this TP.

This line is optional. If it is not included, Communications Server for Linux assumes that the executable file name is the same as the TP name. If you specify a file name with no path, the default path for AIX or Linux systems is **/etc/opt/ibm/sna**; for a Windows client, Communications Server for Linux uses the normal Windows mechanisms for locating the executable file.

ARGUMENTS

Any command-line arguments to be passed to the TP, separated by spaces. These arguments are passed to the TP in the same order as they appear on the command line.

This line is optional. If it is not included, the TP is invoked without any command-line arguments.

TYPE Specify one of the following values:

QUEUED The TP is a queued TP. Any incoming Allocate requests arriving while the TP is running are queued until the TP issues another Receive_Allocate, or until it finishes running and can be restarted. An incoming Allocate request is routed to this TP only if it is received by an LU that is configured to route incoming Allocate requests to this computer.

QUEUED-BROADCAST

The TP is a broadcast queued TP. Any incoming Allocate requests arriving while the TP is running are queued until the TP issues another Receive_Allocate, or until it finishes running and can be restarted. When the TP is started, information about the TP is broadcast to all servers on the LAN; if an LU on another computer receives an incoming Allocate request and has no routing information configured, it can dynamically locate the TP and route the Allocate request to it.

Using QUEUED-BROADCAST instead of QUEUED avoids having to configure explicit routing information for LUs, and enables load-balancing by running more than one copy of the same TP on different computers. However, if you want to avoid broadcasting information in order to reduce LAN traffic, or if you need to ensure that incoming Allocate requests arriving at a particular LU are always routed to the same copy of the TP, you should use QUEUED.

NON-QUEUED

The TP is a nonqueued TP. Communications Server for Linux starts a new copy of the TP each time an incoming Allocate request arrives for it. Do not specify the *TIMEOUT* parameter for a nonqueued TP.

A TP defined as nonqueued cannot be started by an operator; it is always started automatically by Communications Server for Linux. Do not specify NON-QUEUED if the TP is to be operator-started. If a user attempts to start a nonqueued TP, Communications Server for Linux rejects the Receive_Allocate because no incoming Allocate request is waiting for it.

After a nonqueued TP has ended a conversation, it may terminate, or it may issue another Receive_Allocate. For frequently-used programs, this provides a way of avoiding the performance overhead of starting a new instance of the program for each conversation. Each time an Attach is received for a nonqueued, automatically started TP, Communications Server for Linux checks whether there is already a Receive_Allocate outstanding from an instance of this TP. If so, this TP is used for the incoming conversation; otherwise, Communications Server for Linux starts a new instance of the program.

If you use NON-QUEUED, more than one copy of the TP can be running at a time. If the TP writes to files, you need to ensure that different copies of the TP do not overwrite each other's files. To do this, use one of the following methods:

- Ensure that the TP appends data to an existing file instead of creating the file (so that all copies of the TP append data to the same file)
- Design the TP to generate file names at run-time, based on the process ID with which the TP is running (so that each copy of the TP writes to a different file).

This line is optional. If it is not included, or if an invalid value is specified, the default is QUEUED.

TIMEOUT

The maximum length of time, in seconds, that a Receive_Allocate call issued by the TP should block if there is no incoming Allocate request pending. If no incoming Allocate is received in this time, the call fails with a return code indicating "State check - Allocate not pending."

A timeout value of 0 indicates that the call always fails unless an incoming Allocate is already pending when the call is issued. A timeout value of -1 indicates that the call waits indefinitely for an incoming Allocate and does not time out.

This line is optional. If it is not included, or if an invalid value (a non-numeric value) is specified, the default is -1 (infinite).

Do not specify this parameter if the *TYPE* parameter is set to NON-QUEUED. Communications Server for Linux uses a timeout value of 0 for nonqueued TPs, because the TP is always started in response to an incoming Allocate and so there is always one pending.

UNIX

USERID

Specify the user ID that Communications Server for Linux uses to start the TP. The TP is started in the home directory associated with this user ID. This home directory is also the default path for trace files and any other files accessed by the TP (unless the application overrides it by specifying a full path). If the application specifies a file name without a path, Communications Server for Linux searches for the file in this home directory; if the application specifies a file name with a relative path, Communications Server for Linux searches for the file in the specified directory relative to this home directory.

This line is required, and must be specified. The ID must be a valid login ID on the Communications Server for Linux computer; it can be up to 64 characters, unless your AIX or Linux configuration restricts user names to fewer characters.

The executable file for the TP, specified by the *PATH* parameter, must have execute permission for the specified user. In addition, if *USERID* is set to root, the file must be owned by root and must have setuid and setgid permission in order to be started automatically by Communications Server for Linux.

GROUP

Specify the group ID that Communications Server for Linux uses to start

the TP. This must be a valid group ID on the Communications Server for Linux computer; it can be up to 64 characters, unless your AIX or Linux configuration restricts group names to fewer characters.

This line is optional; if it is not included, the default is `other`.

LUALIAS

Specify the local LU alias from which the TP is to accept incoming Attaches.

Note: This parameter can be used only if the TP is an APPC TP. If the TP is a CPI-C application, do not specify this parameter. CPI-C does not support accepting incoming Attaches from a particular local LU; specifying an LU alias (even a blank LU alias) for a CPI-C application will cause errors in routing the incoming Attach to the TP.

This is an eight-character name that must match the name of a Communications Server for Linux local APPC LU.

To indicate that the TP accepts incoming Attaches from any local LU, set this parameter to two double quotation mark characters, `""`, indicating a blank LU alias. If the invokable TP data file contains more than one entry for the same TP name, only one of these entries can specify a blank LU alias; each of the others must specify a different explicit LU alias. Communications Server for Linux matches an incoming Attach for this TP name to a TP specifying the appropriate LU alias, if possible, or to a TP specifying a blank LU alias if no LU alias match can be found.

If a non-blank LU alias is specified in the file, the TP must use the extended form of the APPC RECEIVE_ALLOCATE verb and specify this LU alias as a parameter to the verb. This enables Communications Server for Linux to route the incoming Attach to the correct TP. For more information about the different forms of RECEIVE_ALLOCATE, refer to *Communications Server for Linux APPC Programmer's Guide*. If you need to permit the TP to determine the correct LU alias at run-time rather than building it into the application, you can do this by setting an environment variable to contain the appropriate LU alias (using the *ENV* parameter), and designing the application to read this environment variable in order to determine how to issue RECEIVE_ALLOCATE.

This line is optional; if it is not included, the default is to accept incoming Attaches from any local LU, and the TP can use either form of the APPC RECEIVE_ALLOCATE verb.

ENV Specify any environment variables required by the TP. Each variable is specified in the form *environment_variable=value* on a separate *ENV* line. Up to 64 *ENV* lines can be included; the variables are set in the same order as they appear here.

The string *environment_variable=value* must not contain space or tab characters before or after the = character.

WINDOWS

SHOW

This parameter applies only if the application is a GUI application; it is ignored if the application is a console application. Specify how the application should be displayed when it is started. This parameter is

passed to the application, and not processed by Communications Server for Linux; it is the application's responsibility to interpret it and act on it. You can enter any of the following values:

MAXIMIZED
The application is maximized.

MINIMIZED
The application is minimized.

HIDDEN The application does not appear on the screen.

NORMAL The application is displayed at its normal size and position.

NOACTIVATE
The application is displayed at its normal size and position, and the focus remains on the previously active window. This application's window does not become the active window.

MINNOACTIVATE
The application is minimized, and the focus remains on the previously active window.

This parameter is optional. If it is not included, the default is `NORMAL`.

SECURITY_TYPE
Specify the security type of the TP executable:

APPLICATION
The TP executable is started as an application using the `CreateProcess` system call.

SERVICE
The TP executable is started as a service using the `StartService` system call. In this case, the service must have been previously installed with the Service Control Manager using the name specified by the *SERVICE_NAME* parameter.

This value refers to a TP running as a Windows Service (not to an SNA service TP with a name consisting of 4 characters specified in hexadecimal). Windows allows only one copy of a Service to be running at a time, and so the *TYPE* parameter should not be set to `NON-QUEUED`; if you specify this value, the value `QUEUED-BROADCAST` will be used instead.

SERVICE_NAME
The name of the service installed with the Service Control Manager. This parameter is only used if the *SECURITY_TYPE* is `SERVICE`.

USERID
Specify the domain and user ID that the client should use to start the TP when the *SECURITY_TYPE* is `APPLICATION`. The format for this parameter is *domain_name**user_ID* if the Windows Client computer is part of a domain, or *computer_name**user_ID* (indicating the Windows Client's own computer name instead of a domain name) if the Windows Client computer is not part of a domain.

The client attempts to start the TP in the specified user's logon session. If *USERID* is blank or unspecified, the TP is started in the console session. If the specified user is not logged on, or no user is logged on at the console, the TP is not started and the Communications Server for Linux server is notified of the failure.

Note the following points about the format of these entries:

- You can include a comment line by including # as the first character of the line; Communications Server for Linux then ignores this line. Communications Server for Linux also ignores completely blank lines.
- Each *parameter=value* entry must be on one line; it cannot contain line-break characters. The maximum length of a line is 255 characters; additional characters are ignored.
- White space (space characters and tab characters) at the start or end of a line, or before or after the = character, is ignored (except in the string *environment_variable=value* for the *ENV* parameter).
- Each TP definition begins with the line identifying the TP name, and ends with the end of the file or the next TP name.
- Except for the *ENV* line, which can occur up to 64 times, do not specify the same parameter more than once for the same TP. If you do specify the same parameter more than once, only the last instance of each keyword is used.

Appendix C. 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 p5
System z
System/370
System/390
System 390
VSE/ESA
VTAM
WebSphere
z/OS
z9

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 is a trademark of Intel Corporation.

Linux is a trademark of Linus Torvalds.

Microsoft, Windows, Windows 2003, Windows XP, Windows 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

Numerics

A

B

C

D

E

F

G

H

I

K

L

M

N

P

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
Department AKCA, Building 501
P.O. Box 12195, 3039 Cornwallis Road
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

SC31-6771-02