SunLink™ RJE3770 9.1 Configuration and User's Guide

The Network Is the Computer™

Sun Microsystems Computer Company
2550 Garcia Avenue
Mountain View, CA 94043 USA
415 960-1300 fax 415 969-9131

Part No.: 802-2672-12
Revision A, August 1997

Contents

Figures

Tables

Preface

This manual describes the administration and use of SunLink™ RJE3770, Sun's SNA 3770 (RJE) emulation product. SunLink RJE3770 is a client application of the SunLink 9.1 SunPU2.1 SNA server.

Who Should Use This Book

This book is a reference manual and user's guide for the SunLink RJE3770, the SunLink 3770 Emulation product. System programmers should refer to this manual to install and configure SunLink 3770. Operators should refer to this manual to learn how to invoke the SunLink 3770 emulation capabilities and map the behavior of 3770 devices to the same functions provided by the SunLink RJE3770. The name, *SunLink RJE3770* is used interchangeably with Sun3770 in this manual.

System programmers define the SunLink 3770 emulation devices in the SNA configuration and add these new devices to the Sun PU2.1 SNA server configuration. Usually, one system programmer is responsible for the SNA network updates, while another installs SunLink 3770 on a Unix-based computer and builds the local configuration file. These system programmers must coordinate the parameters chosen for the SNA configuration and the local SunLink 3770 configuration.

As a system programmer responsible for the SNA network configuration, you should be familiar with adding new devices to a VTAM/NCP GEN, determining appropriate log mode tables, and adding JES2 remote definitions.

As a system programmer responsible for the SunLink 3770 installation and configuration on a Unix-based system, you should be familiar with the UNIX® operating system and the configuration of SNA devices.

As a system programmer, you should have background knowledge of the 3770 data entry and display characteristics. You should understand how SunLink 3770 interacts with the SNA network as described in Chapter 1, “Introduction to SunLink RJE3770 9.1."

How This Book Is Organized

This book can be used as a reference for specific installation, configuration, and management procedures. In addition, it can serve as background reading to help you understand the underlying concepts of SunLink 3770 and its operations.

This manual is organized as follows:

Chapter 1, “Introduction to SunLink RJE3770 9.1,” provides a conceptual overview of SunLink 3770 and serves as a background for subsequent chapters.

Chapter 2, “Getting Started,” describes the steps to install, configure, and start SunLink 3770.

Chapter 3, “Using Sun3770,” specifies how to use SunLink 3770 to submit jobs and to manage the 3770 data streams.

Chapter 4, “Input Processing” describes the procedure for creating a Job Control Language (JCL) file, which SunLink 3770 prepends to your data file and transmits to JES2.

Chapter 5, “Output Processing,” describes the steps SunLink 3770 takes when a job request is received from JES2.

Chapter 6, “Automated Processing,” describes how to create Unix shell scripts to automatically submit jobs to JES2.

Chapter 7, “Input Queuing,” describes how to access the SunLink 3770 job input queue for scheduling and submitting jobs.

Chapter 8, “Sun Server Configuration,” specifies the syntax and descriptions of SunLink 3770 configuration directives in the SunPU2.1 SNA Server configuration.

Chapter 9, "SNA Configuration for Sun3770," describes the updates required for the SNA host network configuration.

Chapter 10, "Troubleshooting Sun3770," lists activities that will aid in diagnosing communications problems.

Chapter 11, "Sun3770 Utilities" describes the use of Sun utilities for ASCII /EBCDIC translation.

Appendix A, "Sun3770 Configuration Example," provides an example SunLink PU2.1 SNA Server configuration file that can be used as a basis to configure LUs for a SunLink 3770 SNA Client.

Appendix B, "JES2 Console Commands," lists commonly used console commands.

Appendix C, "Sample JCL Procedure," provides sample procedures for automated job submission and host output post-processing, along with the necessary Job Control Language (JCL).

Appendix D, "ASCII/EBCDIC Translation Tables," describes how to change ASCII/EBCDIC translations; it also includes an example of an ASCII/EBCDIC translation file for Spanish.

Appendix E, "Sun3770 Error Messages," describes the system messages, the Sun basic message display utility, and the structure of a message.

Related Documentation

Sun Documentation

- *SunLink IBM 3270 9.1 End Node Planning and Installation Manual (802-2665)*
- *SunLink SNA PU2.1 9.1 Server Configuration and Administration Manual (802-2673)*

IBM Documentation

- *IBM Systems Network Architecture Concepts and Products, GC30-3072*
- *IBM VTAM Installation and Resource Definition, SC23-0111*
- *JES2, 5740-XYS, Commands, SC23-0048*
- *Introduction to the IBM 3770 Data Communication System, GA27-3144*

- IBM *VTAM Installation and Resource Definition and Component Description (802-2673)*
- *Component Description for the IBM 3776 and 3777 Communication Terminal, GA27-3145*
- *OS/VS2 MVS Utilities, GC26-3902*
- *Sessions Between Logical Units, GC20-1868-2*

Typographic Conventions

The following table describes the typographic changes used in this book.

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

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

Table P-1 Shell Prompts

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

Ordering Sun Documents

SunDocs[SM] is a distribution program for Sun Microsystems technical documentation. Easy, convenient ordering and quick delivery is available from SunExpress. You can find a full listing of available documentation on the World Wide Web: `http://www.sun.com/sunexpress/`

Table P-2 SunExpress Contact Information

Country	Telephone	Fax
United States	1-800-873-7869	1-800-944-0661
United Kingdom	0800-89-88-88	0800-89-88-87
Canada	1-800-873-7869	1-800-944-0661
France	0800-90-61-57	0800-90-61-58
Belgium	02-720-09-09	02-725-88-50
Luxembourg	32-2-720-09-09	32-2-725-88-50
Germany	01-30-81-61-91	01-30-81-61-92
The Netherlands	06-022-34-45	06-022-34-46
Sweden	020-79-57-26	020-79-57-27

Table P-2 SunExpress Contact Information *(Continued)*

Switzerland	0800-55-19-26	0800-55-19-27
Holland	06-022-34-45	06-022-34-46
Japan	0120-33-9096	0120-33-9097

Sun Welcomes Your Comments

Please use the *Reader Comment Card* that accompanies this document. We are interested in improving our documentation and welcome your comments and suggestions.

If a card is not available, you can email or fax your comments to us. Please include the part number of your document in the subject line of your email or fax message.

- Email: `smcc-docs@sun.com`
- Fax: SMCC Document Feedback
  1-415-786-6443

Introduction to SunLink RJE3770 9.1

1

This chapter provides an overview of an IBM SNA 3777 emulation on a Sun3770 workstation. It describes the client-server relationship and Sun3770 components.

1.1 Sun3770 Emulation Overview

Sun3770 emulates an IBM SNA 3777 Model 3 Job Entry station on Sun™ Workstations™, providing Sun users with a Remote Job Entry (RJE) workstation. An RJE workstation can transmit batch jobs to an IBM host for processing and can receive job output from the IBM host. Responsible for interpreting 3770 data streams, the Sun3770 emulator controls the local 3770 console, receives jobs on printer and punch streams, and sends jobs on reader streams. The Sun3770 emulator uses the SunPU2.1 SNA server to establish and control the data streams—LU Type 1 SNA sessions—with mainframe applications.

The physical connection to the SNA network is provided by the SunPU2.1 SNA server. The SunPU2.1 SNA server and the Sun3770 client can either execute on the same system, or be separated and communicate over a TCP/IP network.

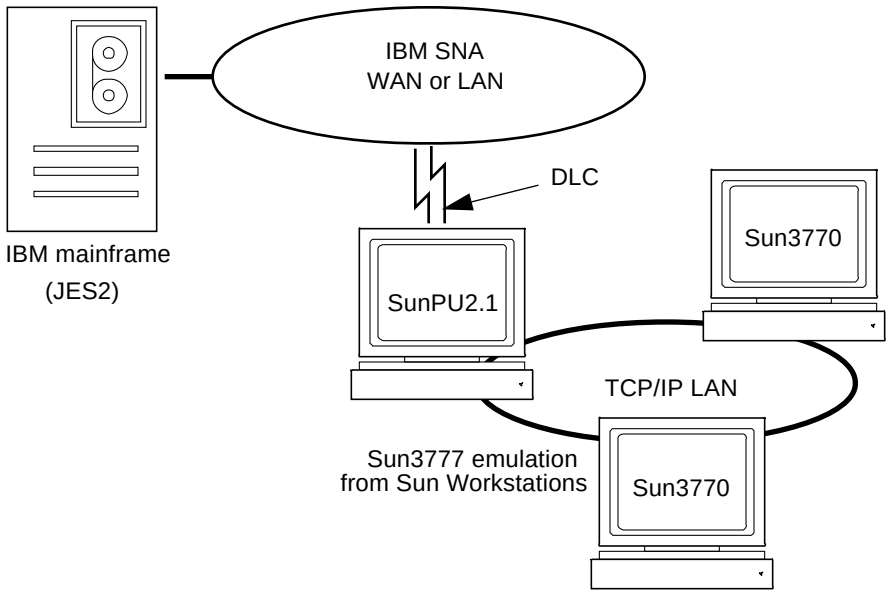

Figure 1-1 Sun3770 and the SunPU2.1 SNA Server

The workstations on the TCP/IP LAN can access the mainframe by invoking the Sun3770 emulator in an open window, and then logging on to the mainframe applications.

1.2 Sun3770 Client

Sun3770 runs as an SNA client program on a Sun Workstation. It attaches to a SunPU2.1 SNA server to gain access to the SNA network. The relationship between the Sun SNA resources and their comparable IBM devices is illustrated in Figure 1-2.

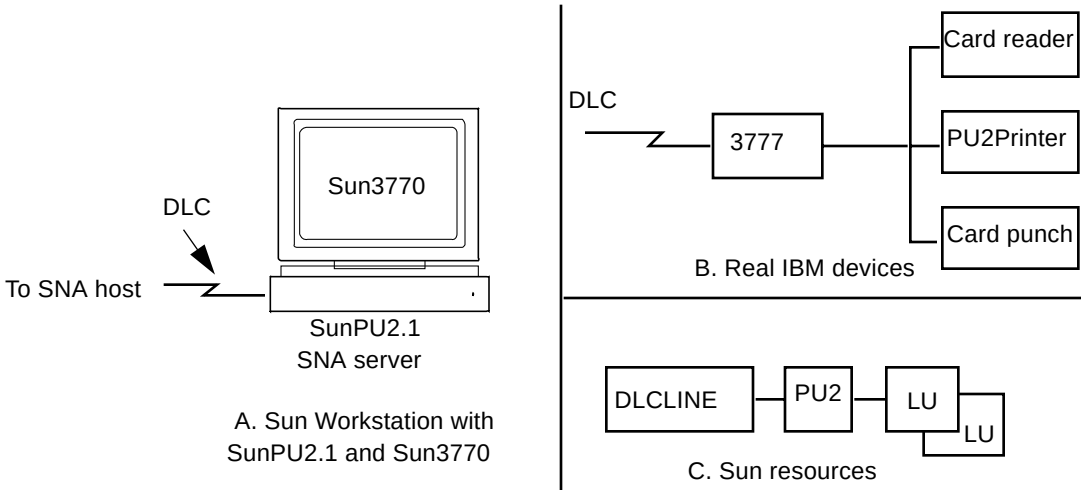

Figure 1-2 Sun3770 and SNA

Sun3770 emulates an IBM 3777 Model 3 Job Entry workstation with its associated card readers, printers, and card punches. The SunPU2.1 SNA server emulates the functions of an SNA Control Unit, for example IBM 3777 Cluster Controller. The relationship of the Sun SNA resources and the IBM devices, as illustrated in Figure 1-2, is discussed below:

- Detail A illustrates the SunPU2.1 SNA server running on a Sun Workstation; the client SNA program, Sun3770, is also running on the Sun Workstation.
- Detail B shows a parallel configuration composed of real IBM devices.
- Detail C relates the Sun SNA resources to the IBM devices, where the SunPU2.1 SNA server represents the PU2 and attaches to the SNA network via a DLC (SDLC, Token Ring, or QLLC). The LUs represent Sun3770.

1.3 *Sun3770 Emulation*

Sun3770 emulates an IBM 3777 Model 3 Job Entry workstation. To provide SNA 3770 emulation, the Sun3770 application controls the screen format, local data files, and the data streams. Each Sun3770 is connected logically to the IBM host by the following types of data streams (up to six per Sun3770):

- Reader streams that carry job input to the IBM host.
- Printer or card punch streams that receive job output from the IBM host.
- Console stream—a bi-directional stream that issues commands and receives messages from the IBM host.

Each stream is associated with an LU and uses an LU-LU SNA session. You can associate a stream with an LU using Job Control Language (JCL) commands or Sun3770 management commands.

1.3.1 Sun3770 Features

Sends jobs to the IBM host

Jobs are transmitted to the IBM host over card reader streams. From the IBM host's perspective, a reader stream is a data transmission path associated with a card reader device.

Receives job output

Output is received from the IBM host over printer and card punch streams. From the IBM host's perspective, printer and card punch streams are data transmission paths associated with printers and card punches.

Compresses and decompresses data

Data sent on card reader streams is compressed. Data compression enables Sun3770 to store consecutive occurrences of the same character in just a few bytes. Data received on printer and card punch streams is also compressed.

Data compression reduces the number of bytes transmitted on a communications line and consequently increases the logical throughput of batch transfers.

Uses multiple logical units (MLUs)

Up to six Logical Units (LUs) are supported. Each LU can be assigned to transfer card reader, card punch, or printer stream data. By supporting MLUs, Sun3770 enables you to execute multiple jobs simultaneously.

Automated processing

OpenWindow™ commands are accepted to submit jobs and manage job output. OpenWindow shell scripts can be used to create customized programs to control Sun3770.

Input job queuing

All input requests to the Sun3770 job queue are enqueued. The job queue allows you to set the priority for an input request, as well as specify the date/time that the job should be submitted to the IBM host. The job input queues are preserved across a system interruption, providing the necessary checks and balances for a completely automated batch environment.

Transparent data

Job data is transferred as transparent data. Use this feature to transfer binary data.

Console commands and messages

A console data stream connects to a job entry subsystem (JES) that resides on the IBM host. You can issue commands to JES and receive messages from JES.

Console logging

All console messages can be logged to a file that you can view later.

Job post-processing

User-supplied program is invoked after receiving job output from the IBM host.

Operator control

Local commands control and monitor the resources associated with the Sun3770 workstation, for example LU status and Job cancel.

Message database

Localized message databases are handled with an on-line help facility.

SNA communications

Communications capabilities are provided by the SunPU2.1 SNA server.

Diagnostic facilities

Extensive tracing, a logical data scope (sunscope), and a local operator (sunop) are available.

1.4 Sun3770 Components

Sun3770 is made up of several different components. Figure 1-3 illustrates the relationship between each Sun3770 component.

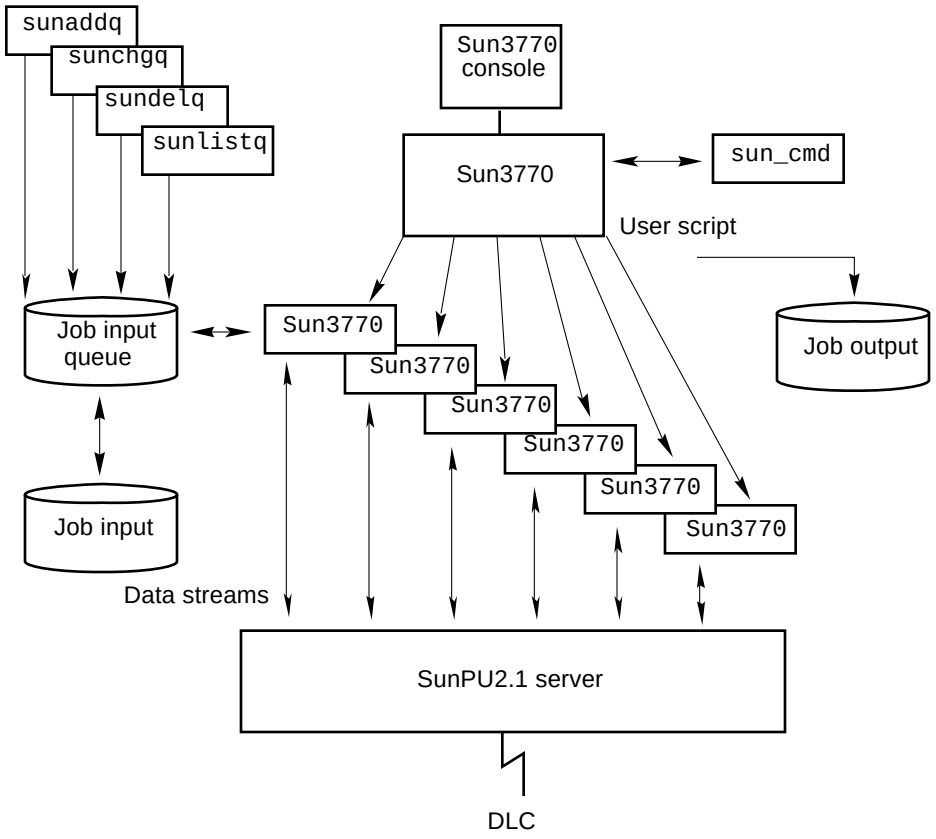

Figure 1-3 Sun3770 Components

1.4.1 Sun3770

Sun3770 controls the submission of jobs to the IBM host. You enter commands for Sun3770 either through the Sun3770 console or via `sun_cmd`. Sun3770 distributes your commands to the appropriate data stream controllers (`sun3770`). Additionally, job sequencing is controlled via job input queues. The

job input queues maintain job sequencing via priority, date and time, and specific LU assignment. The job input queue is also preserved across a system failure.

1.4.2 sun3770

The `sun3770` processes control the 3770 data streams. One `sun3770` process manages the data on one data stream. Each data stream uses one LU port on the SunPU2.1 SNA server.

1.4.3 sunaddq, sunchgq, sundelq, *and* sunlistq

`sunaddq`, `sunchgq`, `sundelq`, and `sunlistq` are Unix applications that enable you to enqueue job input for Sun3770 from Unix command lines or from within Unix shells. The job input queues allow you to prioritize job input as well as specify the date/time that a job should be submitted. In addition, the job input queue is preserved across a system failure. Once a job is queued, it remains in the job input queue until it is successfully submitted to the host or is explicitly deleted by the user. Use these commands to write your own programs to handle Sun3770 job input.

1.4.4 *Job Input*

The job input files contain job control language (JCL) that specifies the actions to be taken by the Job Entry System (JES). Job input can be used to transfer files to the IBM host or to retrieve files from the IBM host.

1.4.5 *Job Output*

The job output files contain the results of a JES output resulting from a previous job submission. For file transfers, Sun provides a program to automatically remove the JCL that controls the transfer of data (`sun_stripper`).

1.4.6 sun_cmd

sun_cmd is a Unix application that enables you to generate Sun3770 commands from either Unix command lines or from within Unix shells. Use sun_cmd to write your own programs to control Sun3770.

1.4.7 User Script

After receiving job output from the IBM host, Sun3770 invokes a post-processing script. You can build your own script to convert job output into the proper format for mail, printers, and other devices.

1.4.8 ASCII/BCDIC Translation

You can control ASCII/EBCDIC translation on both input and output jobs. Additionally, Sun provides ASCII/EBCDIC translation programs suna2e and sune2a. These programs read from standard input and write to standard output using a Unix command pipeline. You also can specify translation tables.

1.4.9 SunPU2.1 SNA Server

The SunPU2.1 SNA server provides Sun3770 client SNA programs access to the SNA network. SunPU2.1 SNA server implements the lower-level SNA protocols and interfaces to the physical connection to the SNA network. Sun3770 uses LU Type 1 ports provided by the SunPU2.1 SNA server.

1

Getting Started 2

This chapter describes the necessary steps for establishing SNA connectivity. It focuses on configuring and starting the SunPU2.1 SNA server, and starting Sun3770.

Use this chapter to get up and running with Sun3770 as fast as possible. To do this, first use a simple configuration of a single-link connection into the SNA network, a single PU2, and a single Sun3770 program. Appendix A lists the sample configurations distributed with Sun3770. Once you are successfully connected to a remote application, you may extend this configuration to meet your specific requirements.

Before Sun3770 can communicate with IBM mainframe applications, you must correctly install, configure, and activate the SunPU2.1 SNA server. Sun delivers the SunPU2.1 SNA server with installation scripts and sample configurations to ensure that you can begin communicating with IBM mainframe applications immediately.

2.1 Installation Summary

An overview of the steps to install your software and license system follows.

Each step is covered in further subsequent sections. After successfully completing these steps, be sure that Sun3770 can access the IBM mainframe and JES2.

1. Verify installation requirements. If you have not read this section, refer to Chapter 1, “Introduction to SunLink RJE3770 9.1” for details.

2. Determine whether you will be accessing a remote or local CD drive.
3. Open the plastic case containing your CD-ROM, place CD-ROM in the caddy, and place the caddy in the CD-ROM drive.
4. Become superuser.
5. Mount the CD-ROM.
6. Install the Sunlink SNA 3770 RJE 9.1 package (`Sunw3770`) on the target client Sun Workstation. This may or may not be the same workstation where your `sunpu2.1` server exists.
7. Install the licensed software package (`SUNWlicsw`) on the Sun Workstation that contains your FlexLM licenses for the SunPU2.1 server software. In most cases this is where your SunPU2.1 server is installed. If using FlexLM license servers, it may be another workstation.
8. Coordinate local and target network configurations.
9. Update the SunPU2.1 SNA server's local configuration (`sunpu2.config`).
10. If your SunPU2.1 is already running, do the following:
    a. Stop the SunPU2.1 server.
    b. Stop the FlexLM license daemon.
    c. Update the license file.
    d. Start the FlexLM license daemon.
11. Start Sun3770.

Figure 2-1 illustrates a sample configuration distributed with Sun3770 (refer to Appendix A); the remaining sections use this sample configuration as an example. Several other sample configurations are included with the SunPU2.1 SNA server as well.

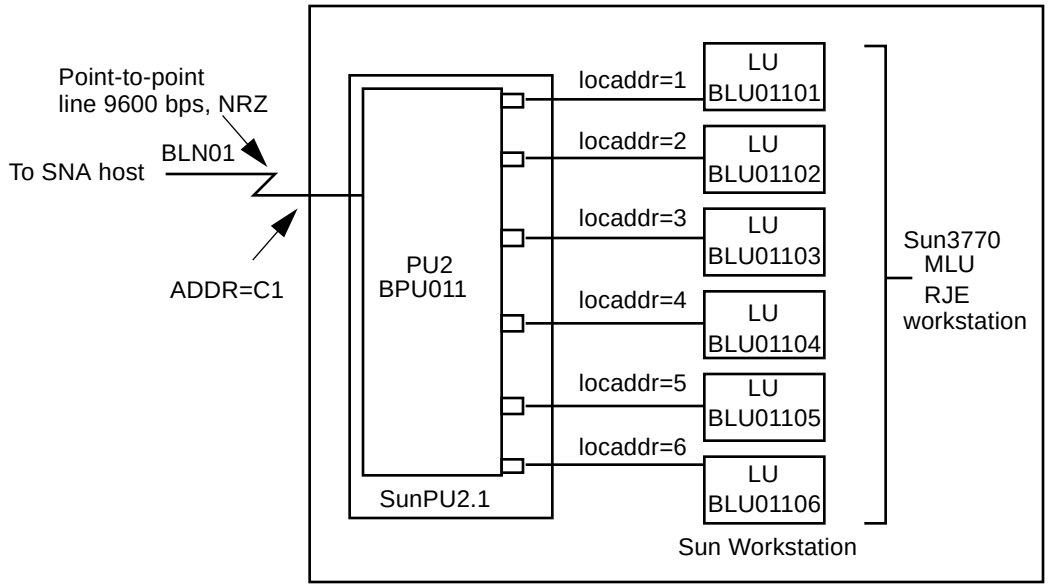

Figure 2-1 Sample Configuration

2.2 *Installing and Mounting the CD-ROM*

This section describes how to insert the CD-ROM in the drive and how to export the CD-ROM if you are using a remote CD-ROM running under Solaris.

The SunLink SNA Peer-to-Peer 9.1 and the SunLink SNA Peer-to-Peer RunTime 9.1 software is distributed on CD-ROM. You must have either a local CD-ROM drive, or a drive that is accessible remotely via an existing network.

Note – Your target machine for *installing* the SunLink SNA Peer-to-Peer 9.0 or SunLink SNA Peer-to-Peer RunTime 9.1 software must always be a system running Solaris 2.3 or Solaris 2.4.

---

1. **Remove the CD-ROM from its plastic case and place the CD-ROM in its caddy (Figure 2-2).**
2. **To add a CD-ROM device in Solaris 2.3 or Solaris 2.4:**
   a. **Connect the device to the machine's SCSI bus.**

b. To a power source, power up the device, then halt your machine by entering `init 0`.

c. At the ok prompt enter `boot -r`.

When the CD-ROM is properly inserted into the caddy, the label is visible.

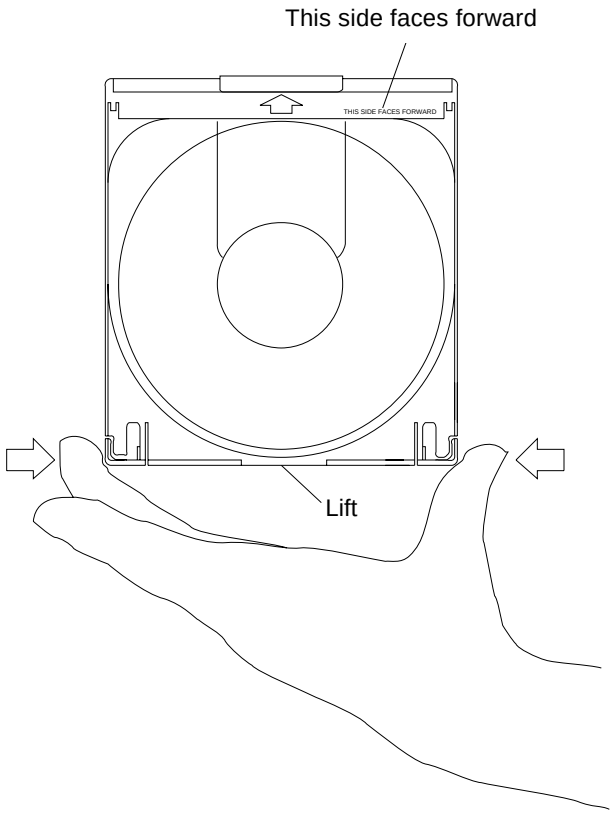

Figure 2-2 Insert the CD-ROM into the Caddy

3. Insert the caddy into the drive slot. Make sure the CD drive is powered on. Refer to Figure 2-3.

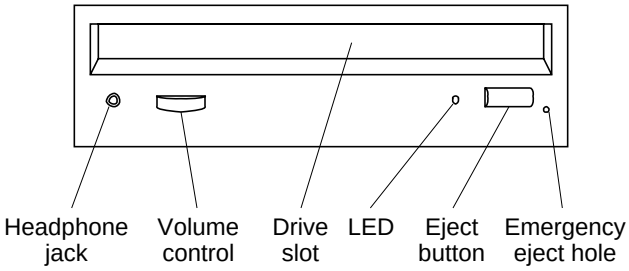

Figure 2-3 CD Device Showing Drive Slot

- Go to “Mounting the CD-ROM from a Local CD Drive” if you are accessing a locally installed CD drive.
- Go to “Mounting the CD-ROM from a Remote CD Drive,” if you are accessing a remotely installed CD drive.

2.2.1 *Mounting the CD-ROM from a Local CD Drive*

Note – You do not have to perform the following steps if you are running the Solaris 2.4 operating environment and have not modified the `/etc/vold.conf` file. If your system meets these two requirements, the CD is mounted automatically.

To mount the CD-ROM from a local CD drive:

1. Become superuser.

```
hostname% su
Password: enter your superuser password
```

2. Create a directory to mount the CD-ROM if it does not already exist:

```
hostname# mkdir /cdrom
```

3. **Mount the CD-ROM:**

```
hostname# mount -r /dev/dsk/c0t6d0s0 /cdrom
```

2.2.2 Mounting the CD-ROM from a Remote CD Drive

To install the software on a machine that does not have its own CD-ROM drive, you must perform activities both on the remote machine (the machine with the CD-ROM device) and on the local machine (the machine onto which you are installing the software). Be sure you follow the directions carefully. The instructions specify the machine on which you must perform the steps.

2.2.3 Exporting from the Remote Machine

Because you are NFS-mounting the software from a remote CD-ROM, you must first export it from the remote machine. The methods are slightly different for remote Solaris 2.3, and Solaris 2.4 systems. However, each system requires that you log in as, or become superuser.

2.2.4 Exporting from a Remote Solaris 2.3 or Solaris 2.4 System

Perform the following steps on the *remote* machine:

1. **On the remote machine, become superuser:**

```
hostname% su
Password: enter your superuser password
```

2. **On the remote machine, make a directory on which to mount the CD-ROM if it does not already exist:**

```
hostname# mkdir /cdrom
```

3. **Mount the CD-ROM on the remote machine:**

```
hostname# mount -r /dev/dsk/c0t6d0s0 /cdrom
```

4. **If the NFS daemons are not already running, start them by entering the following commands:**

```
hostname# /usr/lib/nfs/nfsd 8
hostname# /usr/lib/nfs/mountd
```

5. **Export the directory from the remote machine:**

```
hostname# share -F nfs -o ro /cdrom
```

6. **Check to see that the machine is exporting the directory by entering the command** `share`**. The screen should show the** `/cdrom` **directory:**

```
hostname# share
. . .
-       /cdrom      ro     " "
. . .
```

2.2.5 *Mounting the Remote CD-ROM on the Local Machine*

After you export the remote CD-ROM, you can now NFS-mount it on your local machine. Perform the following steps on the *local* machine:

1. **On the local machine, become superuser:**

```
hostname% su
Password: enter your superuser password
```

2. **Make a directory on which to mount the CD-ROM if it does not already exist:**

```
hostname# mkdir /cd
```

3. **Mount the remote directory on `/cd`:**
   For the Solaris 2.3 or Solaris 2.4 operating environment:

```
hostname# mount -r -F nfs remote_machine:/cdrom/cdrom0 /cd
```

2.2.6 Installing the Software Using pkgadd

Note – If you are installing the SunLink SNA Peer-to-Peer 9.1 packages and the SunLink SNA Peer-to-Peer RunTime 9.1 packages, you should install SunLink SNA Peer-to-Peer RunTime 9.1 packages *first*.

1. **Become superuser.**

```
hostname% su
Password: enter your superuser password
hostname#
```

2. **Invoke pkgadd, specifying the CD-ROM as the distribution device.**
   Use the `-d` option to specify the directory where you mounted the CD-ROM as the source directory for `pkgadd`.

   a. **For Solaris 2.3 and Solaris 2.4 locally installed CD:**

```
hostname# /usr/sbin/pkgadd -d /cdrom/cdrom0
```

   b. **For a remotely installed CD:**

```
hostname# /usr/sbin/pkgadd -d /cd
```

2.2.7 Installing SunLink 3770 RJE 9.1

The following procedure describes how to install the SunLink 3770 RJE 9.1 CD, 3770/RJE, PU2.1 gateway and license package. The steps are listed below.

1. **Install the new SUNW3770 package from the CD.**

```
% cd /cdrom/sunlink_sna_3770_rje_9_0
% pkgadd -d . $cwd
The following packages are available:
1. SUNW3770         3770 Device Emulator
                   (sparc) 9.1
2. SUNWlicsw         FlexLM License System
                   (sparc) 2.0
3. SUNWlit           STE License Installation Tool
                   (sparc) 2.0
4. SUNWpu21          SNA Physical Unit Type 2.1
                   (sparc) 9.1, REV=01
Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]: 4
```

2. **Type the number(s) of the package(s) you want to install or press Return for** `all`**.**

2.2.8 Installing a Remote Sun3770

You can also run Sun3770 on other systems in your network, using the system running the SunPU2.1 SNA server as a gateway to the SNA network. To install a remote Sun3770, move the Sun3770 files to the remote system. The *SunLink IBM 3270 9.1 End Node Planning and Installation Manual* discusses the distribution of Sun3770 programs in your network. When you start a remote Sun3770, you must specify the name of the system running the SunPU2.1 SNA server (use the `-h` *hostname* option).

2.3 Coordinating Configurations

The SNA resources described in the local SunPU2.1 SNA server configuration must mirror the values in the SNA host network configuration. You must coordinate the updates of these two configurations before establishing a connection to the SNA network. Consult your SNA host system programmer to determine the specific values for the parameters outlined below. For specific details on coordinating the SNA configurations, refer to the *SunLink SNA 9.1 PU2.1.0 Server Configuration and Administration Manual.*

Before you establish a connection to the SNA network, you must determine:

- Line attributes (DUPLEX and NRZI parameters of the SDLCLINE directive)
- Line station address (ADDR parameter of the PU2 directive)
- Port number of the logical unit (LOCADDR parameter of the LU directive)
- Logmode table entry for RJE (DLOGMOD parameter of the LU directive)

In addition, the Sun3770 devices need to be added to the devices included in the JES2 remote definitions.

2.4 Configuring SunPU2.1

To configure the Sun3770 SNA connections:

1. Use the configurations in the Sun Installation directory (listed in Appendix A), or another sample configuration included in the *SunLink SNA PU2.1 9.1 Server Configuration and Administration Manual.*

2. Update `sunpu2.config` to reflect your local parameter values.

Particularly relevant parameters and their values in the SunPU2.1 SNA server configuration for the sample configuration are listed in Table 2-1.

Table 2-1 Example Configuration Parameters

SunPU2.1 Directive	SunPU2.1 Parameter	Parameter Value	Description
SDLCLINE	DUPLEX	FULL	Point-to-point line
	LINE	LEASED	Non-switched line
	NRZI	NO	Non-return to zero encoding
	SPEED	9600	Line speed in bits per second
PU2	ADDR	'C1'	Line station address
LU	LOCADDR	1	Logical port for LU
	LUTYPE	1	LU type 1 for RJE LUs
LU	LOCADDR	2	Logical port for LU
	LUTYPE	1	LU type 1 for RJE LUs
LU	LOCADDR	3	Logical port for LU
	LUTYPE	1	LU type 1 for RJE LUs
LU	LOCADDR	4	Logical port for LU
	LUTYPE	1	LU type 1 for RJE LUs
LU	LOCADDR	5	Logical port for LU
	LUTYPE	1	LU type 1 for RJE LUs
LU	LOCADDR	6	Logical port for LU
	LUTYPE	1	LU type 1 for RJE LUs

2.5 Starting SunPU2.1

After coordinating the SNA host network configuration and the local SunPU2.1 SNA server configuration, you are ready to start the SunPU2.1 SNA server. The *SunLink SNA PU2.1 9.1 Server Configuration and Administration Manual* lists the options for invoking `sunpu2.1`. (See Chapter 10, “Troubleshooting Sun3770” regarding SNA connectivity problems.)

The `sunpu2.1` reads the local configuration file to learn about its SNA resources during initialization.

To invoke the SunPU2.1 SNA server with the default configuration, `sunpu2.config`:

1. **Become superuser.**

2. **Enter:**

```
# sunpu2.1 &
```

The `sunpu2.1` daemon will terminate immediately if an error is detected during initialization.

3. **To activate a different configuration, enter:**

```
# sunpu2.1 -f config_file &
```

If initialization is successful, `sunpu2.1` responds as follows:

```
PU200001 : Initializing SunPU2.1 SNA Server
PU200002 : Initialization Complete
```

4. **Use `sunop` to display the status of the SunPU2.1 SNA server, including LU status.**

The `sunop` application prompts you for management requests (-> is the `sunop` prompt).

5. Enter `dis` (display status) to view the PU2 status.

```
-> dis

(2) dis

->

OP200025 : (2) Link BLN01 - (2) Active
OP200020 : (2)   Physical Unit BPU011 - (7) Active
OP20002a : (2)     Logical Unit BLU01101 - (2) Active
OP20002a : (2)     Logical Unit BLU01102 - (2) Active
OP20002a : (2)     Logical Unit BLU01103 - (2) Active
OP20002a : (2)     Logical Unit BLU01104 - (2) Active
OP20002a : (2)     Logical Unit BLU01105 - (2) Active
OP20002a : (2)     Logical Unit BLU01106 - (2) Active
```

- If the SNA host successfully connects to the PU2, the Physical Unit status is "Active."
- If the SNA successfully connects to the LU, the Logical Unit status is "Active."

2.6 Starting Sun3770

After you successfully connect to the SNA network, you can start Sun3770.

To invoke Sun3770,

1. Type:

```
% sun3770
```

For a more detailed discussion of all options, see Chapter 3, which lists all the start-up options for Sun3770.

The IBM banner is displayed in the Sun3770 window. If no banner is displayed, refer to Chapter 10, "Troubleshooting Sun3770."

2. Log onto the JES2 application.

(Sample login request:)

```
01 sscp "L APPLID(JES2) LOGMODE(BATCH) DATA(RMT1)"
```

where,

`01` is the LU stream

`sscp` is a request on the SSCP-LU session

`JES2` is the application

`BATCH` is the logmode

`RMT1` is the JES2 remote definitions for your Sun3770

After logging into JES2, you can enter commands.

3. **To obtain the status of all the devices associated with your Sun3770 from JES2, enter:**

```
1 appl “$DU,RMT1”
```

where,

`01` is the LU stream

`appl` is a request on the LU-LU session

`RMT1` is the JES2 remote definitions for your Sun3770.

2.7 Changing SunPU2.1 Configuration

To change the SunPU2.1 SNA server configuration:

1. **Update the** `sunpu2.config` **file.**
2. **Stop the SunPU2.1 SNA server.**
3. **Restart it again.**

The SunPU2.1 SNA server accesses only the `sunpu2.config` file during initialization.

2.8 Stopping SunPU2.1

To stop the SunPU2.1 SNA server,

- **Load the utility** `sunop` **then enter the** `KILL` **command at the prompt.**

2

Using Sun3770 3

The Sun3770 emulator is a client Unix application. When the Sun3770 emulator is started, it attempts to connect up to six (6) LU Type 1 ports on a SunPU2.1 SNA server. The LUs are either selected by the caller, or the first available LUs are connected.

3.1 Invoking Sun3770

♦ **To invoke Sun3770, enter the following command line:**

```
sun3770 [ -h host_name ] [ -p pu_name ] [ -P | -A|a file_name ]
        [ -s number_of_sessions ] [ -S spool_directory ]
         [ -C activate_command ] [ -D deactivate_command ]
         [ -v ] [ -d ] [ -t -i ]
```

Sun3770 starts one Sun3770 emulator process. This is a full function SNA 3770 emulator.

3.2 Command Line Options

The optional keywords on the command line are explained in Table 3-1.

Table 3-1 Command Line Options

Option	Description
`-h host_name`	Sun3770 requests that the LU services manager connect to the SunPU2.1 server located on the named Sun host. If the `-h` option is omitted, the local Sun host is called.
`-p pu_name`	Sun3770 requests that the PU server connect to LUs belonging to the named physical unit. The PU must be configured in the local configuration file. If the `pu_name` option is omitted, the PU server may select any available LUs.
`-d`	Sun3770 becomes a daemon process and is detached from the controlling terminal. When this option is requested, invoke Sun3770 as a background process (i. e., `Sun3770 -d >& log.file &`).
`-a file_name`	Identifies the file containing the ASCII/EBCDIC translation table. (See Appendix D.)
`-A file_name`	Identifies the file containing the ASCII/ EBCDIC translation table to use, and requests that Sun3770 print the tables during initialization. (See Appendix D.)
`-P`	Requests Sun3770 to print the ASCII/EBCDIC translation tables during initialization. (See Appendix D for the file format.)

Table 3-1 Command Line Options *(Continued)*

Option	Description
`-c activate_command`	Specifies a command to execute on LU activation. Sun3770 will `fork(2)` and `execlp(2)` a Bourne Shell (`sh(1)`) with the initial command argument (`-c`) set to the value of this string. The meta character `%s` is expanded into the session number before the shell is forked. This option may be used to implement an automatic logon.
`-D deactivate_command`	Specifies a command to execute on LU deactivation. Sun3770 will `fork(2)` and `execlp(2)` a Bourne Shell (`sh(1)`) with the initial command argument (`-c`) set to the value of this string. The meta character `%s` is expanded into the session number before the shell is forked.
`-s sessions`	Specifies the number of sessions that Sun3770 should start. This value may specify from 1 to 6 sessions. The default value is 6.
`-S spool_directory`	Specifies the directory for the Sun3770 job input queue. This option is required if you are running multiple Sun3770s from the same directory, as each daemon uses hidden files in the spool directory. The default value is the current directory.
`-t -1`	Trace mode of operation. The trace files are created in `/tmp` with the names `sunlib_pid` and `sunlib_pid.1`. Traces accumulate in the `sunlib_pid` file until 1000 trace points are logged. Then, this file is renamed `sunlib_pid.1` and tracing continues in `sunlib_pid`.

3.3 Child Processes

At start-up, Sun3770 spawns up to six (6) child processes, corresponding to sessions 1 through 6. Communication to the child processes is via TCP/IP sockets. The TCP/IP port number is assigned by the system and are contained

in file bx_7, in the current working directory. The user invoking Sun3770 must have write permissions to the current working directory in order to create the bx_7 file.

- To run Sun3770 in the background, you must use the `-d` option to force Sun3770 to detach from the controlling terminal.
- To run Sun3770 with fewer than six (6) LUs, use the `-s` option to specify the number of sessions (LUs) to start.
- To run multiple copies of Sun3770 (more than 6 LUs), each copy of Sun3770 must start in a different directory. This is because Sun3770 creates several files to keep track of its queues and links.

3.4 Remote Commands

Remote commands send messages to the SSCP or to host application programs, such as JES2, or cause a modification in the flow of data to and from the host.

Remote processing is done with the remote commands `appl`, `cncl`, `inpt`, and `sscp`. All of these commands cause or modify a data transmission between the IBM host and your Unix system.

3.4.1 `appl`

Name

`appl`

Sends a message to the JES2 application.

Synopsis

```
nn appl “message”
```

Description

`nn`

Identifies the destination session id (e.g., 1 - 6) or all available sessions ('*').

`“message”`

This is the message sent to the IBM host application. The maximum message length is 80 characters. If “message” contains embedded quotation marks, use single quotation marks within “message” and double quotation marks to enclose it, or vice versa.

For example: “hello 'world' “ or 'hello "world" .'

Example:

```
1 appl "$d all"
```

Request on session 1 that the host systems application (in this case, JES2) display the host system's active job status on your terminal.

3.4.2 cncl

Name

cncl

Requests to cancel data transmission.

Synopsis

```
nn cncl
```

Description

Requests that the host cancel the data transmission to or from the application. Once the host cancels the transmission, all resources associated with the transmission are available for use with other transmissions. cncl does not cancel the session.

nn

Identifies the destination session id (e.g., 1 - 6) or all available sessions ('*').

Example:

```
1 cncl
```

The host cancels the outbound data transmission directed to the Sun3770 logical print-format disk file associated with the session.

3.4.3 inpt

Name

inpt

Requests that a job be sent to the host.

Synopsis

```
nn inpt [-ms medium/subaddress] [-trn] [-xlt] [-ll line_len]
file_name
```

Description

inpt

Requests that the supplied file be submitted to the host application. inpt sends a job to the host application.

nn

Identifies the destination session id (e.g., 1 - 6) or all available sessions ('*').

-ms medium/subaddress

Identifies a logical address at the IBM host to which the inbound data is sent. The value is a two-character field where the first character is the medium and the second character is the subaddress. Defined medium values are 'C' for card and 'E' for exchange. Defined subaddress values are 0-9, and A-F. The subaddress corresponds to units 0-14, and "any," where value F represents "any." If this option is not entered, the default destination 'C0' is chosen for the current job.

-trn

Identifies the input data as binary data that causes Sun3770 to submit the input data as fixed length card images (using SCS control TRN 0x35). The input data is not interpreted and all records submitted are of the length specified via option -ll (see below).

-xlt

Requests Sun3770 to translate the input data from ASCII to EBCDIC.

-ll line_len

Identifies an optional line length to be specified to the host application. If this option is defaulted (not entered), an 80-character line length is assumed. The maximum value possible is 255.

`file_name`

This is the path name of the input file or device to be used.

Example:

```
2 inpt -ms c1 simplejob
```

Sends the host application on session 2, the job within the file `simplejob` and tells the host the input medium/subaddress is “c1.”

3.4.4 sscp

Name

`sscp`

Sends a message to the System Service Control Point in the IBM mainframe.

Synopsis

```
sscp “message”
```

Description

`nn`

Identifies the destination session id (i.e., 1 - 6) or all available sessions ('*').

`“message”`

This is the message to be sent to the System Service Control Point in the IBM mainframe. Typically, the `message` is a logon or logoff request. The maximum message length is 80 characters. If `“message”` contains embedded quotation marks, use single quotation marks within `message` and double quotation marks to enclose it, or vice versa.

For example: “hello 'world' “ or 'hello “world” .'

Example:

```
1 sscp “l applid(jes2) logmode(batch) data(rmt1)”
```

Sends a message requesting a logon to JES2 (an application program on the IBM host). If the logon is successful, a message acknowledging the successful logon is sent to Sun3770.

3.5 Local Commands

Local commands control activation and deactivation of resources and the flow of data within the computer. Local processing is done using the local commands `outp`, `stts`, `help`, and `quit`. These commands control the routing of data and messages within the systems and provide status information on data transmission and sessions.

3.5.1 `outp`

Name

`outp`

Assigns a medium or subaddress, which will be sent by the host to an output file.

Synopsis

```
nn outp [-xlt] [-ll line_length] [-ms medium/subaddress] [-s]
        [file_name] [-cmd 'command']
```

Description

`nn`

Identifies the destination session id (i.e., 1 - 6) or all available sessions ('*').

`-xlt`

Requests the Sun3770 to translate the output datastream from EBCDIC to ASCII.

`-ms medium/subaddress`

Identifies a logical address to the IBM host application for direct output. The value is a two-character field where the first character is the medium and the second character is the subaddress. Defined medium values are ‘P’ for print, ‘C’ for card, and ‘E’ for exchange. Defined subaddress values are 0-9, and A-F. The subaddress corresponds to units 0-14, and “any,” where value F represents “any.” If this argument is not entered, data for all valid medium/subaddress combinations is redirected to the named file. If the current `medium/subaddress` combination has already been redirected, that file is closed and the new file is opened. If the subaddress selected is 'F,' all valid medium/subaddress combinations for the specified medium are redirected to the named file.

Note – If the specified medium/subaddress combination is currently receiving a job from the IBM host, the command is rejected.

`-ll line_len`

Identifies an optional line length to be specified. If this option is not entered, an 80-character line length is assumed. The maximum value that may be entered is 255. The value entered overrides the host specification from the FMH1 Begin Destination.

`-s`

The specified output filename is treated as a series filename. The output filename is suffixed with the string “`.m.n`”, where `m` is the session number and *n* is the sequence number of the job received from the host. Sequence numbers start with the value one (1).

`file_name`

path name of the output file or device to be used.

`-cmd 'command'`

The supplied command is passed as the initial command to a Bourne shell (`/bin/sh`) on receipt of a complete job (Function Management Header 1, end destination) from the host. The command is *not* validated for Bourne shell syntax. Keywords specified within the command string are replaced by their associated values at command initiation. Keywords are identified by the string %keyword. See Table 6-2 for a description of all valid keywords. To specify the character % as part of the command string, specify the string %%.

Example:

```
03 outp -ms p1 simplejob.output
```

This assigns output for session number 3 — medium / subaddress “p1” to a file, `simplejob.output`.

3.5.2 `stts`

Name

`stts`

Requests status of the local session and resources.

Synopsis

```
nn stts
```

Description

`stts` displays the current status of an inbound or outbound data stream, a session, the media addresses, and the general status of Sun3770.

`nn`

Identifies the destination session id (e.g., 1 - 6) or all available sessions ('*').

Example:

```
03 stts

B3770003d: (3)----------------------------------------

B3770003e: (3)   Status for PU:BPU011 - LU:BLU01103

B3770003f: (3)   Session States: SSCP-LU:CONTENTION LU-
LU:INACTIVE
```

```
B37700040: (3)   RU sent      = 0

B37700041: (3)   RU received = 0

B3770003d: (3)----------------------------------------
```

3.5.3 help

Name

help

Requests help information about the Sun3770 application commands.

Synopsis

```
help [sscp|appl|cncl|stts|inpt|outp|quit]
```

Description

The help parameter displays extensive information about the specified Sun3770 command.

3.5.4 quit

Name

quit

Requests termination of the Sun3770 application.

Synopsis

```
quit
```

Description

The quit parameter causes the session and all resources to be deallocated, and the product to terminate.

Example:

```
quit
Good bye
(%)
```

3.5.5 `flus`

Name

`flus`

Requests Sun3770 to flush the output file.

Synopsis

`quit`

Description

The `flus` parameter causes the current output files to be flushed. If Sun3770 output is redirected to a file, these files can grow very large and correlation of requests and responses is more difficult. `flus` clears the output files, releasing file system space.

Example:

```
flus
```

Input Processing 4

Sun3770 sends a file to the host computer with a minimum of effort on your part. In fact, you may automate this procedure (see Chapter 6). Sun3770 handles all necessary SNA protocol exchange with the host computer, packages the input file data into SNA Request Units (RUs), and translates the input file data from ASCII to EBCDIC (on request).

4.1 Processing and Set Up

The amount of preprocessing and set-up depends upon your application. At a minimum, you must perform the following steps:

1. Construct an input file, which includes:
   - Leading and opening JCL
   - Input data (if any)
   - Trailing and closing JCL (optional)
2. Log on to the host computer.
   Do this through the Sun3770 `sscp` command. Successful “log on” to host computer binds the Sun3770 LUs in session with the host application (for example, JES2).
3. Redirect the host output (optional).

Do this through the Sun3770 `outp` command. All host computer output data is written to a permanent file on your Unix system. If host output is not selected, all output from the IBM host is directed to the current user out (`stdout`).

4. Submit the job.

Do this through the Sun3770 `inpt` command or job input queue command `sunaddq`. The contents of your file are sent to the host computer.

Your file is submitted to the host computer via the Sun3770 `inpt` command or job input queue command `sunaddq`. On receiving the `inpt` command, Sun3770 queues your request on the job input queue. Periodically, the Sun3770 scans the job input queue for pending requests. The Sun3770 validates the input request for the following reasons:

- The requested date or time has arrived.
- The requested LU is bound in session with a primary LU on the host computer (e.g., JES2).
- The requested LU is idle, no current input or output actions are in progress.

Assuming that minimum requirements are met, Sun3770 begins packaging the contents of the input file into SNA RUs for transmission to the host computer. With a non-transparent data stream, Sun3770 packages the data into *variable* length logical records. With a transparent data stream (`-trn`), Sun3770 packages the data into *fixed* length logical records.

4.2 Non-transparent Input

Non-transparent data is submitted to the IBM host as *variable* length logical records with a maximum of *line length* bytes. If translation is requested (`-xlt`), each character read from the input file is translated to EBCDIC. The character is then placed in an in-memory image of the input record. This process continues until an EBCDIC IRS (0x1E) character is encountered, or the maximum *line length* is reached. When the entire input record is assembled, the record is converted into SNA character string control (SCS) format.

The SCS control format conversion includes:

- Stripping trailing space (0x40) characters
- Compressing contiguous identical character sequences (if allowed)
- Delimiting card images by EBCDIC IRS (0x1E)

RUs sent to the host computer contain an integral number of SCS controls. The RUs contain as many SCS controls as possible without exceeding the RU length constraint (established at BIND).

The process of sending RUs to the host computer continues until End of File (EOF) is encountered on the input file

4.3 Transparent Input

Transparent data (`-trn`) is submitted to the IBM host as *fixed* length logical records of *line length* bytes. If translation is requested (`-xlt`), each character read from the input file is translated to EBCDIC. The character is then placed in an in-memory image of the input record. This process continues until the maximum *line length* is reached. When the entire input record is assembled, the record is converted into SNA character string control (SCS) format. The SCS control format conversion includes:

- Prefixing card images with SCS control TRN (0x35)
- Reading *line length* bytes of data from the input file and placing in an in-memory card image
- Stripping off trailing space (0x40) characters
- Compressing contiguous identical character sequences (if allowed)
- Delimiting card images by EBCDIC IRS (0x1E)

RUs sent to the host computer contain an integral number of card images. The RUs contain as many card images as possible without exceeding the RU length constraint (determined at BIND).

The process of sending RUs to the host computer continues until EOF is encountered on the input file. The last card image read from the input file, if less than *`line length`* bytes, is padded out with spaces (0x40). The padded spaces are then stripped off before sending the card image to the host computer.

4

Output Processing 5

Sun3770 handles all of the necessary SNA protocol exchanges with the host computer, extracts your data from SNA Request Units (RUs), and translates the output file data from EBCDIC to ASCII (on request).

Your file is received from the host computer when host output is queued for your remote station and you are bound in session with the primary LU (JES2). The host queues output for your station due to job execution that specifies your station as the destination.

The host computer output data is displayed at the console or redirected to an output file (refer to the `outp` command), as it is received. The basic data handling differs for non-transparent and transparent data streams.

5.1 Non-Transparent Output

Non-transparent data is received from the host computer as EBCDIC data. Each character received from the host computer is treated according to the active SNA character string control (SCS) and is translated to ASCII. The EBCDIC IRS character (0x1e) is translated into an ASCII new line character (0x10). This process continues until the end of the host computer output.

5.2 Transparent Output

Transparent data is output from the host computer as EBCDIC JCL followed by binary data. The JCL is treated exactly as non-transparent data. This continues until the first transparent SCS control is encountered. From this point on, the remainder of the host output is treated as 80 column *fixed length* card images of binary punch data with the following characteristics:

- Card images are delimited by EBCDIC IRS (0x1e).
- The compressed card image is placed in memory.
- The SCS controls are expanded.
- The card image is padded to 80 bytes with space (0x40) characters. The last card image is not padded.
- The resulting card image is written to the user-specified output file.

Automated Processing 6

Sun3770 can operate in an automated mode where command input to the Sun3770 operator console is program-generated. Host output post-processing is invoked when the user exits.

6.1 Automated Input

Sun3770 operator commands can be submitted to Sun3770 via the sun_cmd application. All Sun3770 operator commands may be submitted via sun_cmd.

Invoke sun_cmd is with the following syntax:

```
sun_cmd [-d directory_name] command
```

where

-d directory_name

is the name of the directory in which Sun3770 was started. Sun3770 and sun_cmd communicate via a TCP/IP socket. The port number with which Sun3770 communicates is in file bx_7. The directory name specified identifies where this file is located. If this option is not specified, sun_cmd will look on the search path ($PATH),

where

command

is any Sun3770 operator command. If the command has more than one word, it must be quoted. If the command contains quotes (`sscp, appl,"` enclose the command string within single quotes.

Operator commands received by Sun3770 from `sun_cmd` are identified on the console output as:

```
Script Input: command
```

6.2 User Exit

After Sun3770 successfully receives the host computer output file, Sun3770 causes the user to exit. The user exit happens only if the host computer output was redirected to a file, via the `outp` command.

The name of the user exit process is determined by the transparency mode of the output file designation as described in Table 6-1.

Table 6-1 User Exit Process

User Exit Name	Transparency Mode
`job_end_script`	Non-transparent output
`trn_job_end_script`	Transparent output

Sun3770 creates a process with the arguments shown in Table 6-2.

If the `outp` command specified the `-cmd` option, Sun3770 will invoke the Bourne shell (`/bin/sh`) with the specified command. The keywords specified in the command are substituted with their corresponding values.

Table 6-2 User Exit Parameters

Parameters	Keyword	Value	Description
1	`%f`	`file name`	The path name of the file given in the `outp` command
2	`%m`	`medium`	The medium code specified by the host Card: `C` Printer: `P` Exchange: `E`

Table 6-2 User Exit Parameters *(Continued)*

Parameters	Keyword	Value	Description
3	%s	subaddress	The subaddress code selected by the host Values are 0x00 through 0x0f
Note: The following parameters are taken from the Peripheral Device Information Record (PDIR). This record may not be provided by your host. If no PDIR is received, the following parameters are *not* provided to the user exit process.			
4	%d	date	Date of data set creation in ASCII characters Format: MM/DD/YY Default: 00/00/00
5	%t	time	Time of data set creation in ASCII characters Format: HH.MM.SS Default: 00.00.00
6	%F	form	Form name in ASCII characters, maximum length 8 characters; default: spaces
7	%C	fcbname	Form Control Buffer (FCB) name in ASCII characters, maximum length 8 characters; default: spaces
8	%t	train	Printer train name in ASCII characters, maximum length 8 characters; default: spaces
9	%c	copies	Copies to produce in ASCII characters, maximum value 8-digit decimal number.
10	%v	volio	Volume of I/O, specifying the number of card images; maximum value, 8-digit decimal number
11	%j	jobname	Host job name in ASCII characters, maximum length, 8 characters; default: spaces
12	%S	stepname	Host job step name in ASCII characters, maximum length, 8 characters; default: spaces
13	%p	procstep	Host procedure step name in ASCII characters, maximum length, 8 characters; default: spaces
14	%D	ddname	Host DD name in ASCII characters, maximum length, 8 characters; default: spaces
15	%N	spinno	Host spin number in ASCII characters, maximum length, 8 characters; default: spaces

Sun3770 uses `execlp(2)` to create this process. Consequently, the user exit procedures may be located anywhere along the search path (`$PATH` environment variable).

Typical job post-processing may include stripping the leading and trailing JCL and distributing the output file to the appropriate destinations. See Appendix C for examples of job post-processing.

6.3 LU Logging

The Sun3770 console and LU logging may be invoked by redirecting the standard output and standard error output at the Sun3770 process initiation. All console output is directed to standard error output for the process. All LU output (that is not redirected to a file via the `outp` command) is directed to standard output for the process. Console output and LU output is differentiated by Sun3770, so separate logs may be maintained for each type of output.

Examples:

```
Sun3770 -d > & log &
```

- Starts Sun3770 as a daemon process, redirecting both console and LU output to `file log`.

  ```
  sh -c "Sun3770 -d 1>lu.log 2>console.log &"
  ```

- Invokes a Bourne shell and starts Sun3770 as a daemon process redirecting console output (`stderr`) to file `console.log` and LU output (`stdout`) to file `lu.log`.

6.4 Samples

Sun provides sample procedures for automated job submission and job post-processing, as well as sample JCL. The samples are listed in Table 6-3, and are also included in Appendix C.

To facilitate stripping the leading JCL from transparent job output, Sun provides a sample program. The C language source code for this application is included with the distribution file, `sun_stripper.c`. This program may require modification to meet the requirements of your host site. However, it will serve as a base for this function.

Table 6-3 Sample Procedures

Procedure Name	Function
`job_get`	Retrieves non-transparent job from the host
`job_put`	Sends non-transparent job to the host
`job_end_script`	Performs non-transparent output post-processing
`trn_job_get`	Retrieves transparent job from the host
`trn_job_put`	Sends transparent job to the host
`trn_job_end_script`	Transparent output post-processing

6.4.1 Example 1

A host job runs week nights and produces a report to be distributed to your Unix system users.

1. Build a shell script to perform the following functions:

```
sun_cmd '* outp /my-dir/my-file'
```

This command redirects host output to a file.

```
sun_cmd '* sscp “l applid(jes2) logmode(batch) data(rmt1)”'
```

This executes the logon command to the host computer.

2. Build a shell script, named `job_end_script`, to do the following:

- Strip the leading or trailing JCL.
- Mail the resulting file to required users.
- Remove the output file.

- Enter the `sun_cmd '* logoff,"` which causes the user to log off from the host computer.

3. Update `crontab(1)` to invoke the `job_submission` shell week nights.

6.4.2 Example 2

A critical database is to be saved on the host computer system.

1. Build a shell script to perform the following functions:

a. `tar(1)` and `compress(1)` the database files.

b. Concatenate the leading JCL and the compressed tar file into

```
/my-dir/trn-job
```

c. Execute this command to redirect host output.

```
sun_cmd '* outp -trn /my-dir/my-file'
```

This command redirects host output to a file.

d. Execute the logon command.

```
sun_cmd '* sscp "l applid(jes2) logmode(batch) data(rmt1)"'
```

This causes logon to the host computer.

e. Execute this command to submit transparent job.

```
sun_cmd '1 inpt -trn /my-dir/trn-job'
```

2. **Build a shell script, named** `trn_job_end_script`**, to perform the following functions**:

   a. **Remove the** `output` **file and the** `trn-job` **file.**

   b. **Execute** `logoff` **command**:

```
sun_cmd '* logoff'
```

   This will cause the user to log off from the host computer.

3. **Update** `crontab(1)` **to invoke the** `job_submission` **shell as required.**

6

Input Queuing 7

Sun3770 provides the capability to queue input requests to be submitted at a specific date and time. Input requests submitted to a busy session are implicitly queued. The queue is managed via commands to add, delete, and list queue contents.

7.1 sunaddq

Sun3770 input requests may be queued for submission to the Sun3770 via sunaddq. Invoke sunaddq with the following syntax:

```
sunaddq [-txcrv] -f file-name [-l lu-name] [-d date-time]
        [-p priority] [-s spool-directory] [-L length]
        [-m medium-subaddress]
```

The command line options are described in Table 7-3.

Table 7-1 `sunaddq` Command Line Options

Command Line Option	Description
`-f file-name`	The file name containing the data to be sent to the host. This parameter is required.
`-l lu-name`	The desired LU that the input request is queued for. If this option is not specified, the first available LU is used.
`-d date-time`	The desired date and time at which the job should be submitted. The format of this parameter is as follows: `[mmdd]HHMM[yy]` refer to `date(1)`. Date and time prior to the current date and time is interpreted as the current date and time. If this option is not specified, the current date and time is used.
`-s spool-directory`	The desired spooling directory read by the Sun3770. If this option is not specified, the current directory is used.
`-p priority`	The desired priority for this job within the input queue. Priorities range from zero (lowest) to nine (highest). Only the superuser can request priority greater than zero. If this option is not specified, priority zero is assigned.
`-L length`	The desired input record length, refer to the `inpt` command syntax.
`-m medium-subaddress`	The desired medium-subaddress for job submission, refer to the `inpt` command syntax.
`-t`	The desired transparency indicator, refer to the `inpt` command syntax.
`-x`	The desired translation indicator, refer to the `inpt` command syntax.
`-c`	Instructs `sunaddq` to `copy` the `-f` *`file-name`* argument to the *`spool-directory`*.
`-r`	Instructs Sun3770 to remove the `-f` *`file-name`* argument following successful job submission. Successful completion of `sunaddq` is indicated via a zero exit status. `sunaddq` outputs the job identifier within the queue. The identifier is used with `sundelq`, and is displayed via `sunlistq`. The identifier is also output on the Sun3770 console, messages indicating that the job was submitted.

7.1.1 sunaddq *Error Codes*

-1 Invalid usage

-2 Queue lock contention

-3 Queue data file error on open

-4 Queue data file error on close

-5 Queue index file error write

-6 Queue data file error write

-7 Queue index file error read

-8 Queue full

7.2 sunchgq

Sun3770 queued input requests may be changed via sunchgq. The effective user id for sunchgq is used to validate permission to delete the queue. Only the superuser may change queue entries submitted via another user id. The attributes of the queue entry that may be changed include the *queue status, date/time,* and *priority.* If you desire to change other queue entry attributes, use the command sundelq to delete the queue entry followed by sunaddq to resubmit it.

Invoke sunchgq with the following syntax:

```
sunchgq -i id-number [-s spool-directory] [-p priority]
        [-d date/time] [-S status]
```

The command line options associated with sunchgq are described in Table 7-3.

Table 7-2 sunchgq Command Line Options

Command Line Option	Description
`-i id-number`	The queue entry identifier that specifies the job to be changed in the queue. This parameter is required.
`-s status`	The queue entry status is updated to reflect the desired state. Possible values are `Q` for queued and `H` for hold. Queued entries are processed when an LU is available and all submission criteria are satisfied. Hold entries are not processed but are held in the queue.
`-s spool-directory`	The desired spooling directory read by the Sun3770. If this option is not specified, the current directory is used.
`-d date-time`	The desired date and time that the job should be submitted. The format of this parameter is as follows: `[mmdd]HHMM[yy]` refers to `date(1)`. The date-time prior to the current date-time is interpreted as the current date-time. If this option is not specified, the current date-time is used.
`-p priority`	The desired priority for this job within the input queue. Priorities range from zero (lowest) to nine (highest). Only the superuser can request priority greater than zero. If this option is not specified, priority zero is assigned. Successful completion of `sunchgq` is indicated via a zero exit status. `sunchgq` outputs the job identifier within the queue. If the priority or date/time of job submission is changed, the job identifier within the queue may change. The identifier is used with `sundelq`, and is displayed via `sunlistq`. The identifier is also output on the Sun3770 console, indicating that a job was submitted.

7.2.1 sunaddq *Error Codes*

-1 Invalid usage

-2 Queue lock contention

-3 Queue data file error on open

-4 Queue data file error on close

-5 Queue index file error write

-6 Queue data file error write

-7 Queue index file error read

-8 Queue full

7.3 sundelq

Sun3770 queued input requests may be deleted via `sundelq`. The effective user id during `sundelq` operation is used to validate permission for the queue deletion. Only the superuser may delete queued entries submitted via another user id. If the `-c` option of `sunaddq` was specified, *`file-name`* is removed when the queued entry is deleted.

Invoke `sundelq` with the following syntax:

```
sundelq -i id-number [-s spool-directory]
```

where the command line options are:

`-i` *`id-number`*

The queue entry identifier specifying the job to be deleted from the queue. This parameter is required.

`-s` *`spool-directory`*

The desired spooling directory read by the Sun3770. If this option is not specified, the current directory is used.

Successful completion of `sundelq` is indicated via a zero exit status.

7.3.1 sundelq *Error Codes*

-1 Invalid usage

-2 Queue lock contention

-3 Queue data file error on open

-4 Queue data file error on close

-5 Queue index file error write

-6 Queue data file error write

-7 Queue identifier not found

-8 Queue identifier invalid

-9 Queue index file error read

7.4 sunlistq

Sun3770 queued input requests may be listed via the `sunlistq` application.

`Invoke sunlistq` with the following syntax:

```
sunlistq [-s spool-directory]
```

where the command line options are as follows,

`-s spool-directory`

The desired spooling directory read by the Sun3770. If this option is not specified, the current directory is used.

Successful completion of `sunlistq` is indicated via a zero exit status. The queue listing is in tabular format with the headings defined in Table 7-3.

Table 7-3 sunchgq Command Line Options

Command Line Option	Description
St	The job status and possible values are: Q for queued, A for active, and H for hold.
Id	The job queue identifier. This value is displayed as a decimal number.
Date/Time	The date and time this entry is scheduled for submission. This value is displayed in ctime(2) format. If none was requested, this value is displayed as a space.
Pri	The queue entry priority within the queue. This value is displayed as a decimal number.
User Id	The effective user id that submitted the queue entry.
LU Name	The LU name for which the job is queued. If none was requested, this value is displayed as an * (asterisk)
File Name	The file name containing the job input data. If the file name is longer than 16 characters the display is continued to the next line.
Trn	The value of the transparent indicator. This value is displayed as Yes or No.
Xlt	The value of the translation indicator. This value is displayed as Yes or No.
Len	The value of the input record length indicator. This value is displayed as a decimal number.
MS	The value of the medium-subaddress specified. If none was specified, this value is displayed as a space.
Rmv	The value of the remove indicator. This value is displayed as Yes or No.

7.4.1 sunchgq *Error Codes*

-1 Invalid usage

-2 Queue lock contention

-3 Queue data file error on open

-4 Queue data file error on close

Sun Server Configuration 8

The Sun3770 emulator attaches to LU Type 1 ports on the SunPU2.1 SNA server. Each port represents one LU. Since LUs and PUs are SNA resources, each LU Type 1 port and PU2 must be configured in the local SunPU2.1 SNA server configuration file.

8.1 SunPU2.1 SNA Server Configuration

The SunPU2.1 SNA server configuration file specifies the SNA resources associated with a Unix system. This configuration file is read during the SunPU2.1 SNA server (`sunpu2.1`) initialization. This chapter discusses the LU directive in the SunPU2.1 SNA server; the LU directive specifies SNA parameters for Sun3770. Refer to the *SunLink SNA PU2.1 9.1 Server Configuration and Administration Manual* for further information on the SunPU2.1 SNA server configuration.

The configuration file is ASCII text. Each directive begins with a directive keyword and ends with a semicolon (;). The conventions shown in Table 8-1 are followed when describing each directive's syntax.

Table 8-1 Configuration Syntax

Mnemonic	Symbol	Description
Brackets	[ ]	Optional argument
Or-sign	|	Separates the options in a list
Uppercase	A	Must be entered as shown either in uppercase or lowercase
Lowercase	a	Variable argument, indicates the type of information
Commas	,	Must be entered as shown
Equal	=	Must be entered as shown
Parenthesis	( )	Must be entered as shown
Single quote	'	Must be entered as shown
Slash slash	,”	Indicates a comment; the rest of the line is ignored

There is no restriction on the use of blank lines, tabs, or spaces to separate directives and their arguments. Comments may appear anywhere and are preceded by “/ /”. Refer to Appendix A for an example configuration.

8.2 LU Directive

The LU directive defines the attributes associated with an LU port operating on the specified PU. Absent parameters take default values. See Table 8-3 for details.

Table 8-2 LU Directive Parameters

LU Directive	Description
Syntax	`LU NAME=lu_name, LU_NAME=associated_pu2, [LUTYPE=lu_type,] LOCADDR=n, [PACING=n];`
Restrictions	`lu_name` must be unique. `associated_pu2` must be a `PU2 NAME` argument.
Arguments: `NAME=lu_name`	`lu_name` can be any alphanumeric string up to 8 characters long. White space (tabs, spaces) is not allowed. NAME is used to identify this LU.
• `PU_NAME=associated_pu2`	`associated_pu2` can be any alphanumeric string up to 8 characters long. White space (tabs, spaces) is not allowed. `PU_NAME` is used to identify the PU2 directive associated with this station.
• `LUTYPE=n`	`n` is an integer specifying the type of this LU (range 0-3). Default: 2. Table Table 8-3 shows the LU types and Sun client programs.
• `LOCADDR=n`	`n` is an integer specifying the local address of this LU (Range 1-255).
• `PACING=n`	`n` is an integer specifying the number of requests allowed in a pacing window. Default: 1.

Table 8-3 shows the LU types and Sun client programs.

Table 8-3 LU Types and Sun Client Programs

LU Type	Sun Client Program
0	SunLU0 API Program
1	Sun3770, SunLU0 API Program
2	Sun3270, SunLU0 API Program
3	SunLU0 API Program

Example:

```
LU     NAME=LU01,          ," User defined name

       PU2_NAME=PU01,      ," Name of PU2 this LU is on

       LUTYPE=1            ," LU Type 1 port (Sun3770 port)

       LOCADDR=1,          ," Local address of the LU

       PACING=1;           ," Pacing window
```

SNA Configuration for Sun3770 9

This chapter explains how to configure Sun3770 in the SNA host network configuration. Be sure to coordinate the updates to the local and SNA host network configurations with the SNA host system programmer. The sample configuration described in Chapter 2 relates to the Sun and SNA directives.

This chapter does not discuss all the requirements for extending the SNA host network configuration. Instead, it explains the additions needed to correctly configure Sun3770 in an SNA network. Refer to the *SunLink SNA PU2.1 9.1 Server Configuration and Administration Manual* for more information on configuring the local Line and Physical Unit resources. Refer to the IBM document, *VTAM Installation and Resource Definition and Component Description for the IBM 3776 and 3777 Communication Terminal* for a complete description of SNA host network configuration.

9.1 SNA Configuration

An SNA network is a hierarchical network where many end-point devices communicate with a few host systems. To add a new device to the SNA network, an SNA host system programmer must update the SNA host network configuration (NCP / VTAM GEN).

The NCP/VTAM GEN lists all the SNA resources connected to an SNA communications controller. Four macros define the resources associated with a PU2 device emulated by the SunPU2.1 SNA server. Table 9-1 shows the NCP/VTAM macros. The LU macro defines the characteristics of the Sun3770 attached to the corresponding LU port on the SunPU2.1 SNA server.

Table 9-1 NCP/VTAM Macros

MACRO	Description
GROUP	Specifies certain common characteristics and functions for a group of links and devices
LINE	Represents the physical line connecting to the PU2 device
PU	Represents the physical unit, the PU2 device
LU	Represents a logical unit attached to the PU2 device. An LU provides a port for SNA applications to gain access to an SNA network. Sun3770 is an SNA application.

The LU type is determined by parameters transferred in the SNA BIND message received while the session is being established. The LOGMODE definition for Sun3770 is included later in this chapter.

The sample configuration connects to the SNA network with one point-to-point link over one Physical Unit Type 2 (PU2), and over Multiple LU (MLU) Sun3770 client SNA program (LUs). Figure 9-1 shows an example of an SNA configuration.

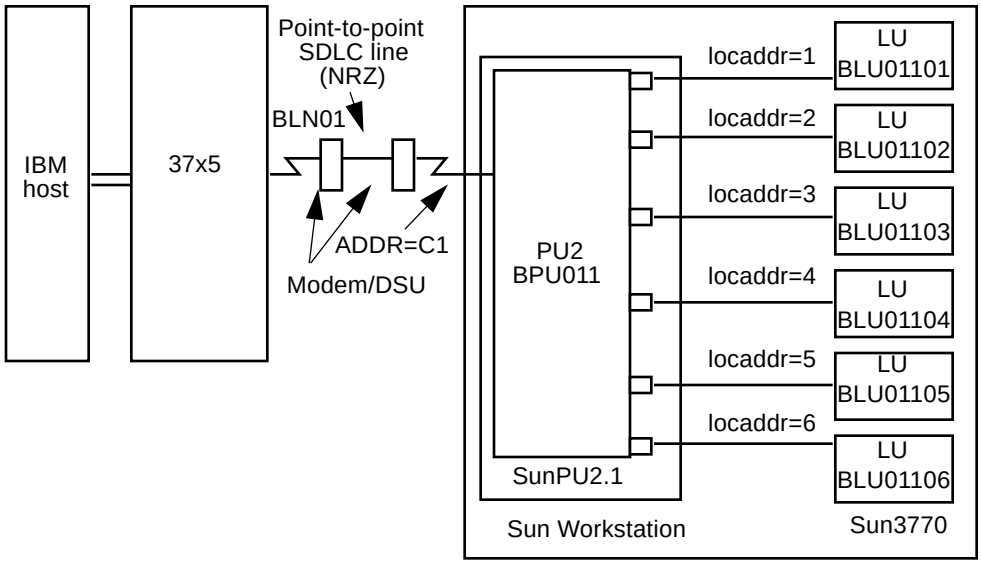

Figure 9-1 Sample SNA Configuration

Together, the SunPU2.1 SNA server and Sun3770 emulate an RJE 3777 workstation (BPU011). Six data streams are configured (BLU01101 through BLU01106). The Unix workstation attaches to the SNA network via a Point-to-Point SDLC line (BLN01). Part of VTAM / NCP GEN for this example is included on the next page.

9.2 Group Macro

Code Example 9-1

```
*****************************************************************
******
*
* GROUP MACRO
*
*****************************************************************
******
GRPSUN0GROUPDIAL=NO,
        LNCTL=SDLC,
        TYPE=NCP,
        ISTATUS=ACTIVE,
*****************************************************************
******
* LINE OPERANDS MOVED UP TO GROUP MACRO
*****************************************************************
******
        CLOCKING=EXT,
        DISCNT=NO,
        SERVLIM=5,
        TRANSFER=9,
        SPDSEL=NO,
*****************************************************************
******
* PU OPERANDS MOVED UP TO GROUP MACRO
*****************************************************************
******
        IRETRY=YES,
        MAXDATA=265,
        MAXOUT=7,
        PASSLIM=11,
*****************************************************************
******
* LU OPERANDS MOVED UP TO GROUP MACRO
```

```
*****************************************************************
******
        MODETAB=ISTINCLM,
        SSCPFM=USSSCS,
        USSTAB=HIS3770,
        PACING=1,
        VPACING=1
*****************************************************************
******
*
* LINE MACRO
*
*****************************************************************
******
BLN01LINEADDRESS=(01,FULL),
        SPEED=9600,
        NRZI=NO,
        DUPLEX=FULL
*****************************************************************
******
*
* PU MACRO
*
*****************************************************************
******
BPU011PUADDR=C1,
        PUTYPE=2
*****************************************************************
******
*
* LU MACROS
*
*****************************************************************
******
BLU01101LULOCADDR=1,DLOGMOD=BATCH
```

```
BLU01102LULOCADDR=2,DLOGMOD=BATCH
BLU01103LULOCADDR=3,DLOGMOD=BATCH
BLU01104LULOCADDR=4,DLOGMOD=BATCH
BLU01105LULOCADDR=5,DLOGMOD=BATCH
BLU01106LULOCADDR=6,DLOGMOD=BATCH
```

9.3 LOGMODE Tables

The VTAM LOGON Mode (LOGMODE) table requires entries to describe the operating characteristics of logical units (LUs). The association between the LU and its LOGMODE entry is created in the LU macro of the VTAM/NCP GEN (or in the definition of a superior resource: GROUP, LINE, or PU).

A LOGMODE entry defines session and presentation values that are included in the session activation request (BIND) sent by the SNA host application to Sun3770. The LOGMODE entry values specify the LU type, maximum message sizes, and session protocol definitions.

Sun3770 emulates an IBM RJE 3777 workstation. The LOGMODE entry that IBM distributes for this type of terminal (BATCH) can be used as the Sun3770 LOGMODE entry. Below is a sample LOGMODE entry.

Code Example 9-2 Sample LOGMODE Entry

```
LOG3770   MODEENT LOGMODE=LOG3770,

                  FMPROF=X'03',

                  TSPROF=X'03',

                  PRIPROT=X'A3',

                  SECPROT=X'A1',

                  COMPROT=X'7080',

                  RUSIZES=X'8686',

                  PSNDPAC=4,
```

```
                SRCVPAC=2,

                SSNDPAC=4,

                PSERVIC=X'01102000F100800000010040'
```

9.4 Application Tables

The JES2 generation defines the remote devices associated with a Sun3770 product. The following example defines the line on which JES2 communicates with Sun3770 (LINE1) and the Sun3770 workstation (RMT1). For the Sun3770 workstation, seven card readers (R1.RD1 to R1.RD7), four printers (R1.PR1 to R1.PR4), and three card punches (R1.PU1 to R1.PU3) are defined.

Note – Check the `&NUMLINES` and `&TPBFSIZ` parameters in the JES2 generation. These parameters set the maximum number of lines and the maximum buffer size for the entire JES2 subsystem.

You can define more than six devices even though the Sun3770 workstation only supports six LUs in the VTAM / NCP generation. You can use the different device emulators with JES2 operator commands and Sun3770 workstation commands as you need them.

Code Example 9-3 JES2 Operator and Sun3770 Workstation Commands

```
LINE1   UNIT=SNA

RMT1    LUTYPE1,COMP,CONSOLE,SETUPINF,BUFSIZE=512,NUMRD=7,NUMPR=4,NUMPU=3

R1.RD1

R1.RD2

R1.RD3

R1.RD4

R1.RD5

R1.RD6

R1.RD7

R1.PR1 CKPTLINE=32767,CDPTPAGE=32767,PRWIDTH=132

R1.PR2 CKPTLINE=32767,CDPTPAGE=32767,PRWIDTH=132

R1.PR3 CKPTLINE=32767,CDPTPAGE=32767,PRWIDTH=132

R1.PR4 CKPTLINE=32767,CDPTPAGE=32767,PRWIDTH=132

R1.PU1 CKPTLINE=32767,CDPTPAGE=32767

R1.PU2 CKPTLINE=32767,CDPTPAGE=32767

R1.PU3 CKPTLINE=32767,CDPTPAGE=32767
```

Troubleshooting Sun3770 10

This chapter provides troubleshooting procedures for Sun3770. Sun provides several tools with Sun3770 to aid in determining and resolving problems.

Note – Troubleshooting information for the SunPU2.1 SNA server and its SNA connections is documented in the *SunLink SNA PU2.1 9.1 Server Configuration and Administration Manual.*

Use these utilities to determine the cause of Sun3770 problems:

- Sun Basic Message Database (`BMD`)
  Documents all errors: describes the cause, effect, and recovery action. Access the messages via the on-line database. The on-line database utility and Sun3770 messages are documented in Appendix D.
- SunPU2.1 Operator (`sunop`)
  Displays status and statistical information of the SunPU2.1 SNA server resources. See the *SunPU2.1 SNA Server Configuration and Administration Manual* for using `sunop`.
- Logical data scope (`sunscope`)
  A data link level scope program showing all transmitted and received frames on SNA data link connections. See the *SunPU2.1 SNA Server Configuration and Administration Manual* for using `sunscope`.
- Tracing
  Sun3770 has extensive internal tracing facilities. The SunPU2.1 SNA server also has extensive tracing capabilities.

10

10.1 On-line Help

The most frequently used tool for troubleshooting is the on-line help facility. All error messages and display information are stored in the Sun Basic Messages Database (BMD).

♦ **To view the message files or to display information for specific messages enter:**

```
bmsg COMPxxxx
```

where COMP represents the first four characters of the message group identifier, and xxxx is the message number.

```
% bmsg B3770002
```

The information displayed about the message describes the meaning of the message and indicates the action you should take. See Appendix E for further information about the on-line help facility.

10.2 Operator Status Information

Use the sunop dis command to determine the status of the Sun3770 session.

♦ **To start sunop, enter sunop followed by dis.**

A sample of operator status is shown in Code Example 10-1.

Code Example 10-1 Operator Status Information

```
% sunop
SunLink Controller
-> dis
(2) dis
->
OP200025 : (2) Link BLN01 - (2) Active
OP200020 : (2) Physical Unit BPU011 - (7) Active
OP20002a : (2) Logical Unit BLU01101 - (2) Active/Session
OP20002a : (2) Logical Unit BLU01102 - (2) Active/Attached
OP20002a : (2) Logical Unit BLU01103 - (2) Active
```

In Code Example 10-1, the first LU (BLU01101) is active and a Sun3770 is in session with a host application. The second LU (BLU01102) is active and a Sun3770 is attached to the LU but a session with a host application does not exist; in this state, Sun3770 usually is displaying the logon banner. The third LU (BLU01103) is active but a Sun3770 is not attached to the LU port.

10.3 Logical Data Scope and Tracing

The `sun3770` process can trace all the messages it sends and receives. Messages are stored as trace points in the `sunlib_pid` file in `/tmp`.

- **To trace all Sun trace messages, start `sun3770` with the trace option `-t -1`.**

Refer to Appendix B for an example `sun3770` trace run. The SunPU2.1 SNA server also has extensive tracing options.

Use `sunscope` to monitor traffic on an SDLC line or an IBM Token Ring network. See the *SunPU2.1 SNA Server Configuration and Administration Manual* for information on `sunscope`.

- **To monitor an SDLC line connected to the synchronous device `zbxa`, enter:**

```
% sunscope -e -d /dev/zbxa
```

- **To monitor an IBM Token Ring connection, where the adapter device is `tr0`, enter:**

```
% sunscope -t -e -d /dev/tr
```

10.4 Common Problems

Table 10-1 summarizes the type of problems you may encounter when running Sun3770. It describes symptoms, probable causes, and corrective actions.

Table 10-1 Common Sun3770 Problems

Symptoms	Probable Cause	Corrective Action
• No login banner	- Unable to connect to SunPU2.1; SunPU2.1 not up	Start SunPU2.1
	- Network not up	Establish network connectivity
	- Unable to connect to SunPU2.1, incorrect *hostname* specified	Invoke Sun3770 with *hostname* of the remote SunPU2.1 system: use `-h` *hostname*
	- Unable to connect to SunPU2.1; services for `brxadmin_pu2` and `brx_pu2_espd` do not match on both systems	Be sure that the `/etc/services` files match on both systems; be sure NIS has updated services
	- SNA Host has not activated the LU	Request SNA system programmer to activate the LU
• SunPU2.1 runs on remote system; unable to ping SunPU2.1 system		
• SunPU2.1 runs on remote system		
• `sunop dis` shows LU Inactive		
• UNBIND from application	Bad session parameters	Use LOGMODE BATCH (or similar LOGMODE)
• 0821 sense code received	Bad session parameters	Use LOGMODE BATCH (or similar LOGMODE)

Table 10-1 Common Sun3770 Problems *(Continued)*

Symptoms	Probable Cause	Corrective Action
• `sun_cmd` fails	Sun3770 not up or `sun_cmd` not executed in same directory where Sun3770 is running	Be sure Sun3770 is up and `sun_cmd` is run in the same directory as Sun3770
• Sun3770 in background does not run when data is entered at keyboard	Sun3770 was not started with the `-d` option	Start Sun3770 with the `-d` option to detach from the terminal when Sun3770 runs in the background
• Error opening files or sockets	Incorrect permissions on directory	Update permissions on the directory to include Sun3770
• `job_end_script` not found	`job_end_script` location not in $PATH variable	Move `job_end_script` to directory in search path or add directory to $PATH variable

10

Sun3770 Utilities *11*

Sun3770 provides utility programs for code set conversion between ASCII and EBCDIC. These utilities read from standard input (`stdin`) and write to standard output (`stdout`), so they may be used in a Unix command pipeline. In addition, Sun3770 provides full control over the conversion tables.

11.1 suna2e

`suna2e` accepts ASCII input and writes EBCDIC output. `suna2e` reads from standard input (`stdin`) and writes to standard output (`stdout`).

♦ **Invoke `suna2e` with the following syntax:**

```
suna2e  [ -P | -A|a file_name ]
```

where the command line options are as follows,

`-a` *`file_name`*

Identifies the file containing the ASCII to EBCDIC translation table. See Appendix D for the file format.

`-A` *`file_name`*

Identifies the file containing the ASCII to EBCDIC translation table and requests that `suna2e` print the tables during initialization. See Appendix D for the ASCII to EBCDIC translation for the file format.

`-P`

Requests that `suna2e` print the ASCII to EBCDIC translation tables during initialization. See Appendix D for the file format.

11.2 sune2a

`sune2a` accepts EBCDIC input and writes ASCII output. `sune2a` reads from standard input (`stdin`) and writes to standard output(`stdout`).

♦ **Invoke `sune2a` with the following syntax:**

```
sune2a  [ -P | -A|a file_name ]
```

where the command line options are as follows:

`-a` *`file_name`*

Identifies the file containing the ASCII to EBCDIC translation table. See Appendix D for the file format.

`-A` *`file_name`*

Identifies the file containing the ASCII to EBCDIC translation table, and requests that `sune2a` print the tables during initialization. See Appendix D for the file format.

`-P`

Requests that `sune2a` print the ASCII to EBCDIC translation tables during initialization. See Appendix D for the file format.

Sun3770 Configuration Example A

This appendix contains an example SunLink PU2.1 SNA server configuration file that can be used as a basis to configure LUs for a Sun3770 SNA client. The sample configuration can be used with minimum changes. You should only need to update the PU2.1 ADDR argument. See the SunPU2.1 SNA server documentation for details on the configuration parameters.

Figure A-1 shows the point-to-point line configuration.

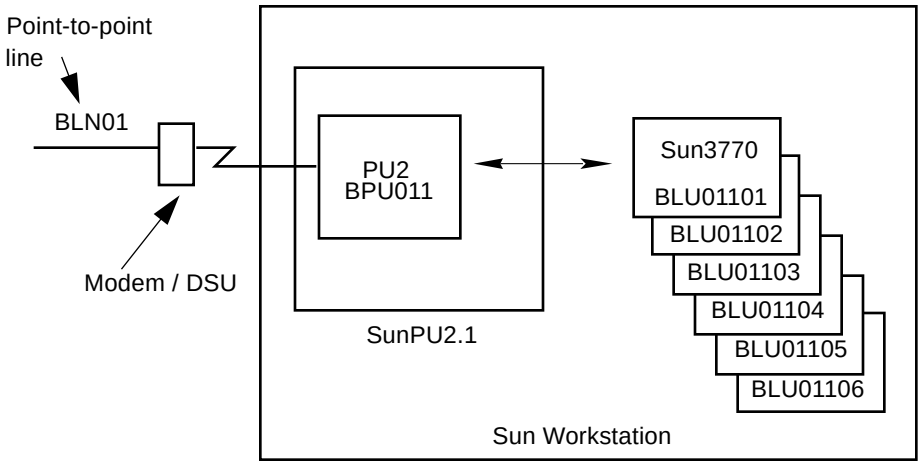

Figure A-1 Point-to-Point Line Configuration

A

Code Example A-1 Sample of SNA Server Configuration

```
// SunLink SunPU2.1 SNA Server Sample Configuration

//

// This sample configuration configures one point-to-point

// Line (BLN01); one Physical Unit (BPU011); and six Logical Units

// (BLU01101-BLU01106).

//

// The physical connection is realized via a 9600 bits/sec

// synchronous modem. The Unix system is connected to the modem

// using serial port A (running the Sun synchronous driver,

// zbxdrv).

//

// You will need to configure the following parameters:

// - PU2 ADDR

//

// Remember:

//SDLCLINE <--> VTAM LINE macro

//PU2 <--> VTAM PU macro, where PUTYPE=2

//LU <--> VTAM LU macro

//
```

```
SDLCLINE   NAME=BLN01,                    // User defined name (8 char max)
           Device='/dev/zbxa',            // Unix device for this line (port A)
           CLOCK=external,                // Clocking by modems (external source)
           DUPLEX=full,                   // Full duplex line (point-to-point)
           LINE=leased,                   // Point-to-point line
           NRZI=no,                       // NRZ encoding
           PAUSE=1,                       // Wait for input during poll cycle (100 ms)
           SPEED=9600;                    // Line Speed (in bits/sec)

PU2        NAME=BPU011,                   // User defined name (8 char max)
           LINK_NAME=BLN01,               // Line name this station is on
           ADDR=x'c1',                    // Station address
           MAXDATA=265,                   // Max data size for frame on link
           OUTFRAME=7,                    // Max outstanding frames allowed
           MODULO=8,                      // SDLC is modulo 8
           ACTIVITY_TIMEOUT=0,            // Host activity timer -- none (in sec units)
           RETRIES=10;                    // Max retransmissions
```

```
LU          NAME=BLU01101,                  // User defined name (8 char max)
            PU_NAME=BPU011,                 // PU2 this LU belongs to
            LUTYPE=1,                       // LU type 1 (3770)
            LOCADDR=1,                      // Local address of LU
            PACING=1;                       // Pacing window


LU          NAME=BLU01102,
            PU_NAME=BPU011,
            LUTYPE=1,
            LOCADDR=2,
            PACING=1;


LU          NAME=BLU01103,
            PU_NAME=BPU011,
            LUTYPE=1,
            LOCADDR=3,
            PACING=1;


LU          NAME=BLU01104,
            PU_NAME=BPU011,
```

```
            LUTYPE=1,

            LOCADDR=4,

            PACING=1;


LU          NAME=BLU01105,

            PU_NAME=BPU011,

            LUTYPE=1,

            LOCADDR=5,

            PACING=1;


LU          NAME=BLU01106,

            PU_NAME=BPU011,

            LUTYPE=1,

            LOCADDR=6,

            PACING=1;
```

A

JES2 Console Commands

This appendix lists some commonly used JES2 console commands. The command format is given in the Sun3770 syntax where nn specifies the stream (1-6 , or '*' for all) and `appl` requests the following command to be sent to JES2.

Code Example B-1 JES2 Console Commands

```
nn appl "$AJ43"                Release job number 43

nn appl "$BPR2"                Backspace PR2 one page

nn appl "$CJ5"                 Cancel job number 5

nn appl "$CRD1"                Cancel activity on RD1

nn appl "$D'MYJOB'"            Display job MYJOB

nn appl "$DA"                  Display active jobs

nn appl "$DN"                  Display jobs in queue

nn appl "$DU,RMT1"             Display RMT1 information

nn appl "$DU,PR1"              Display PR1 information
```

```
nn appl "$EPR3"                Restart PR3

nn appl "$FPR1"                Forward PR1 one page

nn appl "$H'MYJOB'"            Hold job MYJOB

nn appl "$IPR4"                Interrupt PR4

nn appl "$SPU1"                Start PU1

nn appl "$PPU1"                Stop PU1

nn appl "$TPR1,F=XYZ"          Change PR1-forms type to XYZ
```

Sample JCL Procedure

C

Included in this appendix are sample procedures for automated job submission and host output post-processing, as well as the required Job Control Language (JCL). The files included are listed in Table C-1 along with a brief functional description.

C.1 Control Flow of Events

Table C-2 and Table C-3 describe the control flow of events for job submission and output post-processing.

Table C-1 Sample Procedure List

Procedure Name	Description
`jcl_get`	Simple JCL to retrieve a print file that was previously saved in a host data set.
`jcl_put`	Simple JCL to send a print file to the host and save it in a data set.
`job_end_script`	Procedure to handle host output post-processing; this procedure is based on the name of the output file (input parameter). If the output results from procedure `job_put`, invoke procedure `job_get`. Otherwise, the output is from procedure `job_get`, so validate the results.
`job_get`	Shell script to setup output file redirection and submit job `jcl_get`.

Table C-1 Sample Procedure List *(Continued)*

Procedure Name	Description
`job_put`	Shell script to save the original data file (used for results validation). Construct job input file (wrap JCL around data), output file redirection, and submit input job.
`trn_jcl_get`	Simple JCL to retrieve a punch (binary) file that was previously saved in a host data set.
`trn_jcl_put`	Simple JCL to send a punch (binary) file to the host and save it in a data set
`trn_job_end_script`	Procedure to handle host output post-processing. This procedure is based on the name of the output file (input parameter). If the output results from the procedure `trn_job_put`, invoke procedure `trn_job_get`. Otherwise, the output is from the procedure `trn_job_get`, so validate the results.
`trn_job_get`	Shell script to set up output file redirection and submit job `trn_jcl_get`.
`trn_job_put`	Shell script to save the original data file (used for results validation), construct job input file (wrap JCL around data), output file redirection, and submit the input job

Control flow for automated job submission and host output post processing for non-transparent data is shown in Table C-2.

Table C-2 Control Flow for Non-Transparent Data

User	Procedure	Sun3770	Host
Invoke procedure: `job_put` LU *data-file*	`job_put:` Save original data-file. Construct job input (using `jcl_put`) output redirection submit job.		
		Output redirection. Submit job to host.	

Table C-2 Control Flow for Non-Transparent Data *(Continued)*

User	Procedure	Sun3770	Host
			Process job and send output to remote.
		Receive output. Invoke `job_end_script`	
	`job_end_script`: Invoke `job_get`.		
	`job_get`: Output redirection, Submit `JCL` `jcl_get`.		
		Output redirection Submit job to host.	
			Process job and send output to remote.
		Receive output. Invoke `job_end_script`.	
	`job_end_script`: Strip JCL from output. Compare saved input to host output and report differences.		

Control flow for automated job submission and host output post processing for transparent data is shown in Table C-3.

Table C-3 Control Flow for Transparent Data

User	Procedure	Sun3770	Host
Invoke procedure: `trn_job_put` LU *binary-file*.	`trn_job_put`: Save original binary-file. Construct job input (using `trn_jcl_put`). Output redirection submit job		
	Receive output. Invoke `trn_job_end_script`	Output redirection. Submit job to host.	Process job and send output to remote.
`trn_job_end_script`: Invoke `trn_job_get`			Process job and send output to remote.
`trn_job_get`: Output redirection. Submit JCL `trn_jcl_get`	Output redirection. Submit job to host.	Receive output. Invoke `trn_job_end_script`	
`trn_job_end_script`: Strip JCL from output. Compare saved input to host output and report differences.			

C.1.1 JCL_GET

```
//SCSSPLIT JOB (your-job),your-id,MSGCLASS=A,CLASS=A
//*
//* @(#)jcl_get1.2 8/10/91
//*
//* This jcl retrieves a print job that was previously saved in
//* a dataset.
//*
//A EXEC PGM=IEBGENER,REGION=64K
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=your-dataset-name,DISP=(SHR)
//SYSUT2 DD SYSOUT=A
//SYSIN DD DUMMY
```

C.1.2 JCL_PUT

```
//SCSSPLIT JOB  (your-job),your-id,MSGCLASS=A,CLASS=A
//*
//* @(#)jcl_put1.2 8/10/91
//*
//* This jcl sends print data up to the host and saves it in
//* dataset for later retrieval.
//*
//A EXEC PGM=IEBGENER,REGION=64K
//SYSPRINT DD SYSOUT=A
//SYSIN DD DUMMY
//SYSUT2 DD DSN=your-dataset-name,DISP=(SHR)
//SYSUT1 DD DATA
```

C.2 JOB_END_SCRIPT

```
#!/bin/sh
#
# @(#)job_end_script1.2 8/10/91
```

```
#
# This is a SAMPLE job post processing script.
#
# On output completion, when media/subaddress data is redirected to
a file # sun3770 automatically invokes the shell script
"job_end_script".
#
# This script corresponds with the sample job_put script.  It
performs
# the following functions:
# 1. If this is the output from the job_put script, invoke job_get
#    script.
# 2. If this is the output of the job_get script:
#    a.) Strip off JCL.
#    b.) Compares host output to safe-stored input and reports any
#        differences.
# NOTE: sun3770 execlp's process "job_end_script" with the
#       media/subaddress current output file name as the only
argument.
#       This need not be a shell.
echo "job_end_script: starting -- $1"
if [ "$1" = "/tmp/put.job.output" ] ; then
    job_get 1
else
    sed -e "1,/PROCESSING ENDED AT EOD/d" -e "/,$ d" $1 > /tmp/junkz
    sed -e "/ -e "$ d" /tmp/junkz > $1.output
    rm /tmp/junkz
    diff $1.output /tmp/put.data > /dev/null
    if [ $? -eq 0 ] ; then
        echo "     no differences "
        rm $1.output
    else
       echo "    trouble in paradise - diff $1.output /tmp/put.data"
    fi
fi
```

```
echo "job_end_script: complete"
```

C.2.1 JOB_GET

```
#!/bin/sh
#
# @(#)job_get1.1 8/10/91
#
# This is a SAMPLE script to perform automated job retrieval.
# Parameters: 1 - sun3770 session id
#
# 1. Set output file designation via command sun_cmd
# 2. Submit job via command sun_cmd
#
# NOTE: Job post processing is performed via shell script
#       "job_end_script".
#
sun_cmd "* outp  /tmp/get.job"
sun_cmd "$1 inpt jcl_get"
```

C.2.2 JOB_PUT

```
#!/bin/sh
#
# @(#)job_put1.1 8/10/91
#
# This is a SAMPLE script to perform automated job submission.
# Parameters: 1 - sun3770 session id
#             2 - input file name
#
# 1. Copy input file to safe-store file for results confirmation
# 2. Wrap JCL around data
# 3. Set output file designation via command sun_cmd
# 4. Submit job via command sun_cmd
#
```

```
# NOTE: Job post processing is performed via shell script
#       "job_end_script".
#

cat $2        >  /tmp/put.data
cat jcl_put   >  /tmp/put.job
cat $2        >> /tmp/put.job
sun_cmd "*  outp /tmp/put.job.output"
sun_cmd "$1 inpt /tmp/put.job"
```

C.2.3 TRN_JCL_GET

```
//SCSSPLIT JOB  (your-job-name),your-id,MSGCLASS=B,CLASS=B
//*
//* @(#)trn_jcl_get1.2 8/10/91
//*
//* This jcl retrieves a punch job that was previously saved in
//* a dataset.
//*
//A EXEC PGM=IEBGENER,REGION=64K
//SYSPRINT DD SYSOUT=B
//SYSUT1 DD DSN=your-dataset-name,DISP=(SHR)
//SYSUT2 DD SYSOUT=B
//SYSIN DD DUMMY
```

C.2.4 TRN_JCL_PUT

```
//SCSSPLIT JOB  (your-job),your_id,MSGCLASS=A,CLASS=A
//*
//* @(#)trn_jcl_put1.1 8/10/91
//*
//* This jcl sends punch data up to the host and saves it in
//* dataset for later retrieval.
//*
//A EXEC PGM=IEBGENER,REGION=64K
```

```
//SYSPRINT DD SYSOUT=A
//SYSIN DD DUMMY
//SYSUT2 DD DSN=your-data-set-name,DISP=(SHR)
//SYSUT1 DD DATA
```

C.3 TRN_JOB_END_SCRIPT

```
#!/bin/sh
#
# @(#)trn_job_end_script1.2 8/10/91
#
# This is a SAMPLE transparent job post processing script.
#
# On output completion, when media/subaddress data is redirected to a
# file, sun3770 automatically invokes the shell script
# "trn_job_end_script".
#
# This script corresponds with the sample trn_job_put script.  It
# performs the following functions:
# 1. If this is the output from the trn_job_put script,
#    invoke trn_job_get script.
# 2. If this is the output of the trn_job_get script:
#    a.) Strip off JCL.
#    b.) Compares host output to safe-stored input and reports any
#        differences.
# NOTE: sun3770 fork's and execlp's "trn_job_end_script" with the
#       media/subaddress current output file name as the only
argument.
#       This need not be a shell.
echo "trn_job_end_script: starting"
if [ "$1" = "/tmp/trn_put.job.output" ] ; then
    trn_job_get 1
else
    sun_stripper $1 $1.output
```

```
    cmp $1.output /tmp/trn_put.data > /dev/null
    if [ $? -eq 0 ] ; then
        echo "    no differences "
        rm $1.output
    else
        echo "    trouble in paradise - cmp $1.output
/tmp/trn_put.data"
    fi
fi

echo "trn_job_end_script: complete"
```

C.3.1 TRN_JOB_GET

```
#!/bin/sh
#
# @(#)trn_job_get1.1 8/10/91
#
# This is a SAMPLE script to perform automated transparent job
# retrieval.
# Parameters: 1 - sun3770 session id
#
# 1. Set output file designation via command sun_cmd
# 2. Submit job via command sun_cmd
#
# NOTE: Job post processing is performed via shell script
#       "trn_job_end_script".
#
sun_cmd "*  outp -trn /tmp/trn_get.job"
sun_cmd "$1 inpt trn_jcl_get"
```

C.3.2 TRN_JOB_PUT

```
#!/bin/sh
#
```

```
# @(#)trn_job_put1.1 8/10/91
#
# This is a SAMPLE script to perform automated transparent job
# submission.
# Parameters: 1 - sun3770 session id
#             2 - input file name
#
# 1. Copy input file to safe-store file for results confirmation
# 2. Wrap JCL around data
# 3. Set output file designation via command sun_cmd
# 4. Submit job via command sun_cmd
#
# NOTE: Job post processing is performed via shell script
#       "trn_job_end_script".
#

cat $2          >  /tmp/trn_put.data
cat trn_jcl_put >  /tmp/trn_put.job
cat $2          >> /tmp/trn_put.job
sun_cmd "*  outp -trn /tmp/trn_put.job.output"
sun_cmd "$1 inpt -trn /tmp/trn_put.job"
```

C

ASCII/EBCDIC Translation Tables D

You can convert the Sun3770 ASCII to EBCDIC translations. Use the `-A file_name` or `-a file_name` command-line arguments. You can view the current ASCII to EBCDIC translation table via the `-P` command line option.

The format of this file is 512 white-space separated hexadecimal numbers. The '#' character indicates that the remaining part of the line is a comment.

The first 256 characters are used for the EBCDIC to ASCII translation. Sun3770 programs determine the translation of an EBCDIC character by using the character's EBCDIC code point as an index from the start of the first set of 256 characters.

The second 256 characters are used for the ASCII to EBCDIC translation. Sun3770 programs determine the translation of an ASCII character by using the character's ASCII numeric code as an index from the start of the second set of 256 characters. Non-mapped characters in the ASCII to EBCDIC translation table should be mapped to the NUL character (0x00).

D

D.1 Sample ASCII/EBCDIC Translation File for Spanish

```
#
# Spanish Translation Tables
#

#
#            EBCDIC to ASCII
#
#

20 20 20 20 20 20 20 20 20 20 20 20 0a 0d 20 20  #  00-0f
20 20 20 20 20 0a 20 20 20 20 20 20 20 20 20 20  #  10-1f
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20  #  20-2f
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20  #  30-3f
20 20 20 20 E0 E1 20 20 E7 20 5B 2E 3C 28 2B 7C  #  40-4f
20 E9 20 20 E8 ED 20 EF 20 26 5D 24 2A 29 3B AC  #  50-5f
2D 2F 20 20 C0 C1 20 20 20 20 F1 2C 25 5F 3E 3F  #  60-6f
20 C9 20 20 C8 CD 20 CE 20 60 3A BA D1 27 3D 22  #  70-7f
20 61 62 63 64 65 66 67 68 69 20 20 20 20 20 20  #  80-8f
20 6A 6B 6C 6D 6E 6F 70 71 72 AA 20 20 20 20 20  #  90-9f
20 7E 73 74 75 76 77 78 79 7A A1 BF 20 20 20 20  #  a0-af
20 20 20 B7 20 20 20 20 20 20 5E 21 20 20 B4 20  #  b0-bf
7B 41 42 43 44 45 46 47 48 49 20 20 20 F2 F3 20  #  c0-cf
7D 4A 4B 4C 4D 4E 4F 50 51 52 20 20 FC 20 FA 20  #  d0-df
5C 20 53 54 55 56 57 58 59 5A 20 20 20 D2 D3 20  #  e0-ef
30 31 32 33 34 35 36 37 38 39 20 20 DC 20 DA 20  #  f0-ff


#
# ASCII to EBCDIC
#
#
```

```
00 01 02 03 04 05 06 07 08 09 1e 0b 0c 0d 0e 0f  #  00-0f
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f  #  10-1f
40 BB 7F 00 5B 6C 59 7D 4D 5D 5C 4E 6B 60 4B 61  #  20-2f
F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 7A 5E 4C 7E 6E 6F  #  30-3f
7C C1 C2 C3 C4 C5 C6 C7 C8 C9 D1 D2 D3 D4 D5 D6  #  40-4f
D7 D8 D9 E2 E3 E4 E5 E6 E7 E8 E9 4A E0 5A BA 6D  #  50-5f
79 81 82 83 84 85 86 87 88 89 91 92 93 94 95 96  #  60-6f
97 98 99 A2 A3 A4 A5 A6 A7 A8 A9 C0 4F D0 A1 00  #  70-7f
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  #  80-8f
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  #  90-9f
00 AA 00 00 00 00 00 00 00 00 9A 00 5F 00 00 00  #  a0-af
00 00 00 00 BE 00 00 B3 00 00 7B 00 00 00 00 AB  #  b0-bf
64 65 00 00 00 00 00 00 74 71 00 00 00 75 00 77  #  c0-cf
00 7C ED EE 00 00 00 00 00 00 FE 00 FC 00 00 00  #  d0-df
44 45 00 00 00 00 00 48 54 51 00 00 00 55 00 57  #  e0-ef
00 6A CD CE 00 00 00 00 00 00 DE 00 DC 00 00 00  #  f0-ff
```

D

Sun3770 Error Messages

All SunLink IBM Connectivity products are distributed with the BMD message database. All informational and error messages output by Sun3770 are derived from the BMD. Product components running in “user space” access the BMD during runtime. Product components running in “kernel space” access message tables that are constructed at compilation time.

The Sun message handling scheme provides the following advantages:

- All messages output by Sun software are documented. When a message is output, users have a reference that describes the reason why the message was output, its impact, and how to respond.
- On-line message documentation is also available. BMD messages are standard Unix text files and may be browsed using normal system editors and utilities, such as `grep`. Sun also distributes, `bmsg`, that, given a message number, prints the full content of the message.
- Configurable message formats are available. Users are free to modify the message content as needed.
- Consistent message handling is available for all Sun products.

BMD messages for Sun3770s product may be found at the end of this appendix.

E

E.1 Sun Basic Message Display Utility: bmsg

♦ **Invoke bmsg by typing:**

```
bmsg [message_number ," message_number...]] [-v]
```

where:

`message_number` specifies the message to display.

`-v`
requests `bmsg` to print the current version number, and exit.

The `-v` overrides all other options.

If no `message_number` is given, `bmsg` prompts the user and displays the requested message. This continues until the user types `Q` to quit. An example `bmsg` operation follows:

E.1.1 Message Response

ABCD0001

```
MESSAGE: ABCD0001
```

`%s` This is an example message string that starts with a string parameter.

Parameters:

```
1. A string (%s)
```

Cause

This field normally explains what condition(s) causes this message to be displayed.

Effect:

This field normally displays what effect the condition(s) may have had.

Action:

This field normally indicates what actions should be taken when this message is displayed.

```
Please enter message identifier (Q to quit):
Q
%
```

E.2 Sun Basic Message Database

The BMD is a Unix directory that contains several message files, each representing one product component. Each product component owns a range of message numbers and stores message information in its own message file. The file is uniquely named `xxxx_msglib`, where `xxxx` is the product component identifier.

The BMD directory may be located anywhere in the user's search path, as specified by the environment variable *$PATH*. If you want to modify a message file, copy it out of the BMD directory into the current working directory. If *$PATH* includes the current directory "**.**", the message file in the current working directory will be accessed rather than the version in the BMD directory.

E.3 Message Files

A message library is a text file containing multiple message entries. Each message has up to seven defining fields:

- Message number
- Message string
- Message string parameters
- Message display options
- Cause
- Effect
- Action

Message fields are demarcated by blank lines. Messages are separated by a full line of hyphens ('**-**'). An example (fictitious) message entry follows:

Code Example E-1 Sample Message

```
MESSAGE:

ABCD0002Line %s has reached the retry limit (%d)

PARAMETERS:

1. LINE name (%s)
2. Retry limit (%d)

CAUSE:

The specified LINE has attempted to retransmit a message retry
limit times.

EFFECT:

LINE moves into disconnected state.

ACTION:

1.  Check the configured MAXDATA size for the line. If you are
transmitting oversize frames they will be rejected.
2.  Use the Operator to examine the line statistics. If the line
condition is poor, it may be necessary to increase the retry limit.
    Otherwise, you may need to increase the retry timeout value.
```

The following text describes various fields associated with the message files.

MESSAGE Field

The MESSAGE field contains the message number and its associated string. The message number, for example ABCD0001, is fixed. The message string is similar to a “printf” formatted string. The order of the parameters in the string is fixed. The following parameter insert values are defined:

`%s` Inserts a string (`printf(“%s”)`)

`%d` Inserts a decimal integer (`printf(“%d”)`)

`%x` Inserts a hexadecimal integer (`printf("0x%08x")`)

`,"` Escapes the '%' so this character can be included in string (`printf("%%")`).

The message string can be up to 1016 characters long and cannot contain blank lines.

PARAMETERS Field

The parameters in the message string are described in the PARAMETERS field. Each parameter is defined on one line. This field can be up to 1024 characters and cannot contain blank lines.

OPTIONS Field

The OPTIONS field describes how the messages should be displayed. Options can include:

NO_PRINT	Message
NO_MESSAGE_NUMBER	Do not prefix the message with its number.
PROGRAM_NAME	Prefix the message by the program name.
PROCESS_ID	Prefix the message by the user process id.
TIME	Prefix the message by the date and time.

CAUSE Field

The CAUSE field describes why the message is displayed. This field can be up to 1024 characters and cannot contain blank lines.

EFFECT Field

The EFFECT field describes the effect on the product component due to the condition described by the associated messages. This field can be up to 1024 characters and cannot contain blank lines.

ACTION Field

The ACTION field describes the actions the user or administrator should take when the associated message is displayed. This field can be up to 1024 characters and cannot contain blank lines.

Index

D

E

F