HP 9000 Networking

HP-UX SNAplus2 Documentation Update

HP Part No. J2740-90004
Printed in U.S.A.
E0597

Edition 1

Legal Notices

The information in this document is subject to change without notice.

Hewlett-Packard makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Warranty. A copy of the specific warranty terms applicable to your Hewlett-Packard product and replacement parts can be obtained from your local Sales and Service Office.

Restricted Rights Legend. Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DOD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

HEWLETT-PACKARD COMPANY
3000 Hanover Street
Palo Alto, California 94304 U.S.A.

Use of this manual and flexible disk(s) or tape cartridge(s) supplied for this pack is restricted to this product only. Additional copies of the programs may be made for security and back-up purposes only. Resale of the programs in their present form or with alterations, is expressly prohibited.

This software is based in part on the Fourth Berkeley Software Distribution under license from the Regents of the University of California.

Trademark Notices UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

X Window System is a trademark of the Massachusetts Institute of Technology.

MS-DOS and Microsoft are U.S. registered trademarks of Microsoft Corporation.

OSF/Motif is a trademark of the Open Software Foundation, Inc. in the U.S. and other countries.

Printing History

New editions are complete revisions of the manual. The dates on the title page change only when a new edition or a new update is published.

Note that many product updates and fixes do not require manual changes and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one-to-one correspondence between product updates and manual updates.

Edition 1 May 1997

Preface

The HP-UX SNAplus2 Documentation Update provides information not found in previously published HP documentation on SNAplus2.

This Documentation Update includes the following information:

- Information on installing and managing Windows clients
- Details on TN Server management and configuration
- Changes to **`snapadmin`**
- New modes of operation
- Changes to several APIs

Audience

This manual is intended for use by anyone who uses SNAplus2 and requires the above information.

Related HP Documentation

The following publications are Hewlett-Packard manuals that are related to the HP-UX SNAplus2 product and may be referenced in this document:

- HP-UX SNAplus2 Installation Guide (HP Part No. J2740-90001)
- HP-UX SNAplus2 Migration Guide (HP Part No. J2740-90003)
- HP-UX SNAplus2 Administration Guide (HP Part No. J2740-90002)
- HP-UX SNAplus2 API CSV Programmer’s Guide (HP Part No. J2740-90004).

Contents

1 Installing Windows Clients

Software Requirements 12

Required and Optional Software 12
TCP/IP Stacks 12
3270 Emulation Programs 13

Installing and Using the Client Files 14

2 Managing Windows Clients

Overview of Managing Windows Clients 16

Starting and Stopping a Windows Client 17

Starting a Windows Client 17
Stopping a Windows Client 17

Windows Client Security 18

Windows Client Initialization File: sna.ini 19

sna.ini File Layout 19
sna.ini File Contents 21
Configuration 21
Servers 22
Logging 23
API_tracing 26
MSG_tracing 28
CS_tracing 28
Appl_Name 29
CSV_data 30
LAN Access Timeout 30

Contents

Windows Client Invocable TP Data File: sna_tps.ini 32
sna_tps.ini File Format 34
sna_tps.ini File Contents 34
TPname 35
PATH 35
ARGUMENTS 35
TYPE 36
TIMEOUT 37
SHOW 38

3 TN Server Management and Configuration

TN Server Management and Configuration 40
Introduction 40
TN Server Users 40
Managing TN Server (Overview) 41

define_tn3270_access 42
Supplied Parameters 42
Command Parameter Descriptions 43
Subsection Parameter Descriptions 44
Error Return Codes 45
Using the Telnet Daemon's TCP/IP Port 46

query_tn3270_access_def 48
Supplied Parameters 48
num_entries 48
list_options 48
default_record 49
client_address 49
address_format 49
port_number 50

Contents

Returned Parameters - Summary Information 50
default_record 50
client_address 50
address_format 50
Returned Parameters - Detailed Information 51
default_record 51
client_address 51
address_format 51
port_number 52
description 52
lu_name 52
session_type 52
model_override 52
Error Return Codes 53

delete_tn3270_access 54

Supplied Parameters 54
default_record 54
client_address 55
delete_options 55
port_number 55
Error Return Codes 55

4 Changes to SNAPADMIN

Command Line Help 58

Usage Statement 58
General Help 59
Normal Help 59
Detailed Help 59
Help On Command 60
Normal Help on Command 60

Contents

Detailed Help on Command 60

Additional Status Commands 61

Connectivity Status 61
Dependent LU Status 62
LU6.2 Status 64
DLUR Status 64
Node Status 65
All Status 65

Other Command-Line Usage Enhancements 66

Queries List all Items by Default 66
Hex Field Entry 66
Overwrite Individual Parameters 67

5 Additional Changes

New Modes of Operation 70

Back-Compatibility Mode 70
Standard Mode 71
Application Scheduled Mode 72
Using Xt Scheduling 73
Using select() or poll() 73
Choosing Which Mode to Use 74
Building Applications 75

Changes to HLLAPI 76

Back-Compatible HLLAPI Library 76
Thread-Save and POSIX HLLAPI Library 76

1

Installing Windows Clients

This chapter explains how to install SNAplus2 Windows clients.

Software Requirements

This section describes:

- the software that must be installed on a Windows computer for use with the WinSock Client program
- 3270 emulation programs and TCP/IP stacks that have been tested with the WinSock Client

Required and Optional Software

The following software is required:

- A Windows TCP/IP stack
- The Windows Sockets DLL, WINSOCK.DLL, version 1.1 or later.

The following software is optional, and depends on how the client will be used:

- A 3270 emulation program (if the client is to be used for 3270 emulation).

TCP/IP Stacks

The WinSock Client software has been tested successfully with all of the TCP/IP stacks listed below. Note that this list is not intended to be exclusive; if a particular stack is not listed, this does not imply that it cannot be used, but merely that it has not been tested.

- FTP PC/TCP, version 2.3 for DOS / Windows
- Microsoft (both 16-bit and 32-bit stacks)
- Netmanage Chameleon, version 4 (February 1994)
- Sun PC/NFS Pro, version 2.0
- WRQ Reflection Network Series TCP Connection for Windows, version 4.0
- Trumpet, version 2.0b
- Novell LAN Workplace for DOS, version 4.2

3270 Emulation Programs

The WinSock Client software has been tested successfully with all of the 3270 emulation programs listed below. Note that this list is not intended to be exclusive; if a particular stack is not listed, this does not imply that it cannot be used, but merely that it has not been tested.

- Attachmate Extra!, Release 4
- DCA IRMA Workstation for Windows, version 2.2.0
- Eicon Access, version 3.14.1
- NetSoft DynaComm/Elite, version 3.31 or later
- Walldata Rumba for the Mainframe, version 4.1

Installing and Using the Client Files

Follow this procedure:

1. Put the SNAplus2 server in client server mode.
2. Install the 3270 emulator.
3. Use FTP to copy files from the server **`/opt/sna/wincli`** directory to a single directory on the client.
4. Modify the **`PATH`** environment variable to include this directory.
5. Create **`sna.ini`** in your Windows directory (refer to page 19).
6. Copy the **`*.dll`** files from the client directory (see step 3 above) to the Windows system directory. This might overwrite some files already there from the 3270 emulation installation. Ensure that there are no other copies of the **`WDMOD.DLL`** file from other SNA client installations on the system.
7. Run the **`WNAP.EXE`** file that was copied from the SNAplus2 directory. You must run this before you can use any SNA client functions. You may want to make an entry for it in the startup program.
8. Run your 3270 emulator program.

NOTE: Any **`*.dll`** files installed by the emulator that have SNAplus2 supplied equivalents MUST be replaced by the SNAplus2 version.

2

Managing Windows Clients

This chapter explains how to manage SNAplus2 Windows clients.

Overview of Managing Windows Clients

On a Windows client, the component that handles access to SNAplus2 servers is the Network Access Process (NAP). The NAP must be started before you can use SNAplus2 applications or emulation programs on the client. For more information, see Starting a Windows Client, later in this chapter.

When the NAP is started, the client contacts an SNAplus2 server over the TCP/IP network in order to access SNAplus2 features. You can optionally set up SNAplus2 servers to enforce password checking for Windows Clients, so that the client user must enter the correct password when starting the NAP in order to gain access to the server; for more information, see page 18 .

The operation of the client is also controlled by the following files, which are described later in this chapter:

sna.ini Windows Client initialization file. This file contains information on the following:

- Configuration information specific to Windows Clients
- Servers that the client can access
- Logging and tracing options for applications running on the client
- Additional options for CPI-C and CSV applications running on the client

sna_tps.ini Windows Client invocable TP data file. This file contains information on invocable TPs (APPC or CPI-C) that can run on the client; it is equivalent to the **`sna_tps`** file on a UNIX computer.

Starting and Stopping a Windows Client

Starting a Windows Client

To start the Windows Client, either double-click on the Windows NAP icon, or use the normal Windows "File Run" mechanisms to run **`wnap.exe`**. The client then uses the information in the **`sna.ini`** file to locate an SNAplus2 server.

If the server is set up to validate user names for Windows clients (as described in Windows Client security, later in this chapter), SNAplus2 displays a pop-up message requesting a password. The user must type in a password; SNAplus2 uses this password and the user name configured for the Windows Client to validate that the user is authorized to access the server. If the server is not set up to validate user names, the pop-up message does not appear.

If you want to start the NAP automatically when the Windows system is started, you can include the Windows NAP icon in the "Startup" group, or list it in the [windows] section of the **`win.ini`** file as a program to be started automatically.

Stopping a Windows Client

Before stopping the NAP, ensure that all SNAplus2 applications (3270 and 5250 emulation programs or applications using the SNAplus2 APIs) on the Windows Client have been stopped.

To stop the NAP, click on the Windows NAP icon and choose Close. If any SNAplus2 applications are running, the Close option is grayed; if you are sure you want to stop the NAP, stop the relevant applications before retrying to stop the NAP.

Windows Client Security

SNAplus2 provides a facility for validating the user name and password of any Windows Client attempting to contact a SNAplus2 server. This allows you to ensure that only authorized Windows users are able to access the SNAplus2 system.

By default, Windows Client security is not active, so that any computer with the Windows Client software installed can access SNAplus2 servers. To enable Windows Client security, take the following steps:

1. Agree a user name and password with each Windows Client user who is authorized to access the SNAplus2 system.
2. On the Windows Client computer, define this user name using the **`snauser`** parameter in the [Configuration] section of the **`sna.ini`** file, described later in this chapter.
3. On all servers that this client can access (as specified in the sna.ini file), define this user name and password to the UNIX system as a UNIX user name.
4. After starting the SNAplus2 software on a server, use SAM to enable/disable Windows Client security.

   This action enables/disables Windows Client security on all servers in the SNAplus2 domain; there is no need to repeat it when starting the SNAplus2 software on other servers.

When a Windows Client starts up and tries to access a server on which Windows Client security is enabled, the client software displays a pop-up message requesting a password. This password, and the user name from the sna.ini file, are checked against the user names defined to the UNIX system on the server. If the Windows Client user does not specify a password, or if the user name and password cannot be matched with a UNIX user name and password on the server, the server rejects the client's access attempt.

Windows Client Initialization File: sna.ini

The Windows Client initialization file, sna.ini, contains SNA network information (similar to the information held in the client network data file on UNIX clients), and also some additional configuration information that is specific to Windows clients. This file is set up during the client installation process; it is an ASCII text file that can be modified later as required using a standard text editor.

sna.ini File Layout

The contents of the file are as follows:

```
[Configuration]
snauser = user_name
domain = domain_name
invoked_tps = YES | NO
lan_access_timeout = nn


[Servers]
Server1 = * | servername1
Server2 = servername2
   .
   .
   .
Server10 = servername10


[Logging]
exception_logging_enabled = YES | NO
audit_logging_enabled = YES | NO
log_directory = directory
error_file = error_filename
```

```
backup_error_file = backup_error_filename
error_file_wrap_size = error_file_size
audit_file = audit_filename
backup_audit_file = backup_audit_filename
audit_file_wrap_size = audit_file_size
succinct_errors = YES | NO
succinct_audits = YES | NO


[API_tracing]
file1 = trace_filename_1
file2 = trace_filename_2
flip_size = filesize
truncation_length = length

all_api = YES | NO
appc = YES | NO
cpic = YES | NO
csv = YES | NO
rui = YES | NO
nof = YES | NO


[MSG_tracing]
file1 = msg_trace_filename_1
file2 = msg_trace_filename_2
flip_size = filesize
fmi = YES | NO


[CS_tracing]
file1 = cs_trace_filename_1
file2 = cs_trace_filename_2
flip_size = filesize
trace_flags = DATA | DATAGRAM | ADMIN_MSG | ALL | NONE
```

```
[Appl_Name]
APPCTPN = tp_name
APPCLLU = lu_name


[CSV_data]
CSVTBLG = table_G_filename
```

sna.ini File Contents

The following paragraphs explain the contents of the file. Note that, where a parameter in the file takes the values YES or NO, any string beginning with Y or y is interpreted as YES, and any string beginning with N or n is interpreted as NO.

Configuration

This section of the file contains configuration information for the client.

snauser = user_name indicates the user name of the SNAplus2 user on this client. This name was specified during the client installation. It must match the SNAplus2 configuration and UNIX configuration on servers, as follows:

- If the SNAplus2 system is set up to validate user names for Windows clients (as described in Windows Client security, earlier in this chapter), this name must be defined as a UNIX user name on all servers listed below in the parameters Server1 - Server10 (or on all servers that can respond to UDP broadcasts, if the client uses this method to locate a server).
- If the client will be running 3270 or 5250 emulation, and you want to configure the user explicitly instead of using the default emulator user configuration, this name must be defined as an emulator user name in the SNAplus2 configuration, using the define_emulator_user command. See “Managing 3270 Users”, in Chapter 5 of the current *HP-UX SNAplus2 Administration Guide*.
- If neither of the above conditions applies, this line of the file is optional; if it is not specified, 3270 or 5250 users on the client will use the <DEFAULT> user record, if any, in the domain configuration file.

domain = domain_name indicates the domain name of the SNAplus2 LAN, as specified during the server installation. This line is required.

invoked_tps Specify **invoked_tps = YES** if this client is used to run invoked TPs (APPC TPs that issue RECEIVE_ALLOCATE, or CPI-C applications that issue **`Accept_Conversation`** or **`Accept_Incoming`**). In this case, you may also need to set up an invocable TP data file on this client; see Windows Client invocable TP Data File, later in this chapter, for more information.

Specify **`invoked_tps = NO`** if this client is not used to run invoked TPs.

This line is optional; if it is not specified, the default is YES.

lan_access_timeout Specify the time in seconds for which the TCP/IP connection from the client to a server should be kept active while no applications on the client are using SNAplus2 resources. For more information, see LAN access timeout, later in this section.

The minimum timeout is 60 seconds (lower values is rounded up to 60 seconds). To bring down the TCP/IP connection more quickly, you need to stop the NAP on the client.

To indicate no timeout, so that the TCP/IP connection will be kept active as long as the NAP is running on the client, do not specify this parameter.

This parameter is optional; if it is not specified, the default is no timeout.

Servers

This section of the file contains information on SNAplus2 servers that the client can access.

Server1 To indicate that the client should attempt to contact an SNAplus2 server by using a UDP broadcast message to all computers on its TCP/IP subnet (or on all subnets that it can access, if the client computer contains more than one LAN adapter card), specify " * ". If the client fails to contact a server by this method, it will then try using directed messages to one or more named servers (specified by the following lines of the file).

In situations where the client cannot reach any servers using UDP broadcasts, and must use directed messages, specify the name of the first server it should try to contact. This applies in the following cases:

- When the SNAplus2 LAN spans multiple TCP/IP subnets, and there are no SNAplus2 servers in any TCP/IP subnet that the client can access using UDP
- When UDP support is not installed on the client.

In other cases, the use of UDP broadcasts is optional; to specify that broadcasts should not be attempted, specify the name of the first server instead of " * ".

Server2 - Server10 Specify the names of additional SNAplus2 servers that the client should contact, in order of preference. If the client has tried to contact a server using a UDP broadcast (or has tried to contact the server specified in Server1) but has received no response, it will then attempt to contact the server specified in Server2 using a directed message. If this fails, it will try the server specified in Server3, and so on.

These server names are optional, but provide a backup mechanism if the broadcast method of locating a server fails or if the server specified by Server1 is unavailable.

If the client tries all the servers listed without success, it waits for 30 seconds, and then restarts the process of trying to contact a server (either with UDP broadcasts or with the first server listed).

Logging

This section of the file specifies logging options for the client. For more information on logging, see Chapter 8 of the current *HP-UX SNAplus2 Administration Guide*.

NOTE: If central logging is enabled (using the **`set_central_logging`** administration command), all log messages are written to a central file on a server. In this case, only the **`exception_logging_enabled`** and **`audit_logging_enabled`** parameters from this section are used; the remaining parameters are ignored.

exception_logging_enabled Set this parameter to YES to specify that exception messages are recorded, or NO to specify that exception messages are not recorded.

This parameter is optional; if it is not specified, the Windows Client uses the global settings (specified using **`set_global_log_type`**) to determine whether exception messages are recorded. (The initial default, when the SNAplus2 software is started, is that exception messages are recorded.)

audit_logging_enabled Set this parameter to YES to specify that audit messages are recorded, or NO to specify that audit messages are not recorded.

This parameter is optional; if it is not specified, the Windows Client uses the global settings (specified using **set_global_log_type**) to determine whether audit messages are recorded. (The initial default, when the SNAplus2 software is started, is that audit messages are recorded.)

log_directory The full path of the directory where log files are stored on this client. All the log files and backup log files (specified in the following parameters) are stored in this directory.

This parameter is optional; if it is not specified, the files are stored in the Windows installation directory (typically c:\windows).

error_file Name of the file to which error messages are written. This parameter is optional; if it is not specified, the default is **sna.err**.

To log error and audit messages to a single file, specify the same file name for both this parameter and the audit_file parameter.

backup_error_file Name of the backup error log file. When the error log file reaches the size specified in error_file_wrap_size below, SNAplus2 copies its contents to the backup file (overwriting any existing file), and then clears the error log file.

This parameter is optional; if it is not specified, the default is **bak.err**.

To log error and audit messages to a single file, specify the same file name for both this parameter and the **backup_audit_file** parameter.

error_file_wrap_size The maximum size of the log file specified by **error_file**. When a message written to the file causes the file size to exceed this limit, SNAplus2 copies the current contents of the log file to the backup log file and clears the log file. This means that the maximum amount of disk space taken up by error log files is approximately twice **error_file_wrap_size**.

This parameter is optional; if it is not specified, the default is 10000 bytes. If you are logging error and audit messages to the same file, this parameter must be set to the same value as **audit_file_wrap_size**.

audit_file Name of the file to which audit messages are written. This parameter is optional; if it is not specified, the default is **sna.aud**.

To log error and audit messages to a single file, specify the same file name for both this parameter and the **error_file** parameter.

backup_audit_file Name of the backup audit log file. When the audit log file reaches the size specified in **audit_file_wrap_size** below, SNAplus2 copies its contents to the backup file (overwriting any existing file), and then clears the audit log file.

This parameter is optional; if it is not specified, the default is **bak.aud**.

To log error and audit messages to a single file, specify the same file name for both this parameter and the **backup_error_file** parameter.

audit_file_wrap_size The maximum size of the log file specified by **audit_file**. When a message written to the file causes the file size to exceed this limit, SNAplus2 copies the current contents of the log file to the backup log file and clears the log file. This means that the maximum amount of disk space taken up by audit log files is approximately twice **audit_file_wrap_size**.

This parameter is optional; if it is not specified, the default is 10000 bytes. If you are logging error and audit messages to the same file, this parameter must be set to the same value as **error_file_wrap_size**.

succinct_errors Specifies whether to use succinct logging or full logging in the error log file; this applies to both exception logs and problem logs. Allowed values are:

YES	Succinct logging: each message in the log file contains a summary of the message header information (such as the message number and log type) and the message text string and parameters. To obtain more details of the cause of the log and any action required, you can use the **snaphelp** utility on a HP-UX server.
NO	Full logging: each message in the log file includes a full listing of the message header information, the message text string and parameters, and additional information on the cause of the log and any action required.

This parameter is optional; if it is not specified, the default is taken from the previous **set_global_log_type** command issued to the master server. The initial default, before any **set_global_log_type** command has been issued, is to use succinct logging.

If you are using central logging, note that the choice of succinct or full logging for messages from all computers is determined by the setting of this parameter on the server acting as the central logger; this setting can either be from the **`set_global_log_type`** command, or from a **`set_log_type`** command issued to that server to override the default.

succinct_audits Specifies whether to use succinct logging or full logging in the audit log file. The allowed values and their meanings are the same as for the succinct_errors parameter.

API_tracing

This section of the file specifies API tracing options for applications running on the client. For more information on tracing, see Chapter 9 of the current *HP-UX SNAplus2 Administration Guide*.

file1 The full pathname of the trace file, or of the first trace file if tracing is to two files (see **`file2`** below).

This parameter is required if you want to enable API tracing.

file2 The full pathname of the second trace file. This parameter is optional; to indicate that tracing is to one file instead of two files, do not include this line.

If both **`file1`** and **`file2`** are specified, tracing is to two files. When the first file reaches the size specified by **`flip_size`**, the second file is cleared, and tracing continues to the second file. When this file then reaches the size specified by **`flip_size`**, the first file is cleared, and tracing continues to the first file. This ensures that tracing can continue for long periods without using excessive disk space; the maximum space required is approximately twice the value of **`flip_size`**.

flip_size The maximum size of the trace file. If two file names are specified, tracing will switch between the two files when the current file reaches this size. If only one file name is specified, this parameter is ignored; the file size is not limited.

This parameter is optional; if it is not specified, the default is 100,000 bytes.

truncation_length The maximum length, in bytes, of the information written to the trace file for each message. If a message is longer than this, SNAplus2 writes only the start of the message to the trace file, and discards the data

beyond **`truncation_length`**. This allows you to record the most important information for each message but avoid filling up the file with long messages.

This parameter is optional; if it is not specified, SNAplus2 does not truncate messages (all the data from each message is written to the file).

all_api To trace messages for all APIs, set this parameter to YES; in this case, SNAplus2 ignores the parameters **`appc - nof`** below.

To disable tracing for all APIs, set **`all_api`** and all of the parameters **`appc - nof`** below to NO.

To trace only messages for specific APIs, set **`all_api`** to NO, and use the parameters **`appc - nof`** below to indicate which APIs to trace.

This parameter is optional; if it is not specified, the default is NO.

appc To trace APPC messages, set this parameter to YES; otherwise, set it to NO. This parameter is optional; if it is not specified, the default is NO. If **`all_api`** above is set to YES, this parameter is ignored, and APPC messages are traced.

cpic To trace CPI-C messages, set this parameter to YES; otherwise, set it to NO. This parameter is optional; if it is not specified, the default is NO. If **`all_api`** above is set to YES, this parameter is ignored, and CPI-C messages are traced.

csv To trace CSV messages, set this parameter to YES; otherwise, set it to NO. This parameter is optional; if it is not specified, the default is NO. If **`all_api`** above is set to YES, this parameter is ignored, and CSV messages are traced.

rui To trace LUA RUI messages, set this parameter to YES; otherwise, set it to NO. This parameter is optional; if it is not specified, the default is NO. If **`all_api`** above is set to YES, this parameter is ignored, and LUA RUI messages are traced.

nof To trace NOF messages, set this parameter to YES; otherwise, set it to NO. NOF messages are not used directly by applications on Windows clients, but are used internally by SNAplus2 components in obtaining configuration information. This parameter is optional; if it is not specified, the default is NO. If **`all_api`** above is set to YES, this parameter is ignored, and NOF messages are traced.

MSG_tracing

This section of the file specifies options for tracing on Windows Client 3270 emulation programs. For more information on tracing, see Chapter 9 of the current *HP-UX SNAplus2 Administration Guide*.

file1 The full pathname of the trace file, or of the first trace file if tracing is to two files (see **`file2`** below). This parameter is required if you want to enable message tracing; you also need to set the **`fmi`** parameter.

file2 The full pathname of the second trace file. This parameter is optional; to indicate that tracing is to one file instead of two files, do not include this line.

If both **`file1`** and **`file2`** are specified, tracing is to two files. When the first file reaches the size specified by **`flip_size`**, the second file is cleared, and tracing continues to the second file. When this file then reaches the size specified by **`flip_size`**, the first file is cleared, and tracing continues to the first file. This ensures that tracing can continue for long periods without using excessive disk space; the maximum space required is approximately twice the value of **`flip_size.`**

flip_size The maximum size of the trace file. If two file names are specified, tracing will switch between the two files when the current file reaches this size. If only one file name is specified, this parameter is ignored; the file size is not limited. This parameter is optional; if it is not specified, the default is 100,000 bytes.

fmi To trace 3270 messages, set this parameter to YES; otherwise, set it to NO. This parameter is optional; if it is not specified, the default is NO.

CS_tracing

This section of the file specifies options for Client-Server tracing (tracing on messages between the client and SNAplus2 servers). For more information on tracing, see Chapter 9 of the current *HP-UX SNAplus2 Administration Guide*.

file1 The full pathname of the trace file, or of the first trace file if tracing is to two files (see **`file2`** below). This parameter is required if you want to enable **`Client-Server`** tracing; you also need to set the **`trace_flags`** parameter.

file2 The full pathname of the second trace file. This parameter is optional; to indicate that tracing is to one file instead of two files, do not include this line.

If both **`file1`** and **`file2`** are specified, tracing is to two files. When the first file reaches the size specified by **`flip_size`**, the second file is cleared, and tracing continues to the second file. When this file then reaches the size specified by **`flip_size`**, the first file is cleared, and tracing continues to the first file. This ensures that tracing can continue for long periods without using excessive disk space; the maximum space required is approximately twice the value of **`flip_size`**.

flip_size The maximum size of the trace file. If two file names are specified, tracing will switch between the two files when the current file reaches this size. If only one file name is specified, this parameter is ignored; the file size is not limited. This parameter is optional; if it is not specified, the default is 100,000 bytes.

trace_flags To trace messages of all types, set this parameter to ALL.

To trace only messages of a particular type, set this parameter to one of the following values:

ADMIN_MSG	Internal messages relating to Client-Server topology
DATAGRAM	Datagram messages
DATA	Data messages

To disable Client-Server tracing, set this parameter to NONE.

This parameter is optional; if it is not specified, the default is NONE.

Appl_Name

This section of the file specifies options for a CPI-C application. To set these options for particular applications, include a section in this format for each application; replace **`Appl_Name`** with the application program's executable name (not including the .exe filename extension).

For more information on CPI-C, see the *CPI-C Programmer's Guide*.

APPCLLU The name of the local LU that this application uses.

This parameter is optional; if it is not specified, the application will attempt to use the default LU (the LU associated with a local node's control point).

APPCTPN The TP name of the application. This name is used in log and trace files to identify the application. For an invoked application (one that issues **`Accept_Conversation`**), it is also used to match the TP name on an incoming Allocate request with the correct application; note that the invoked application can also use the **`Specify_Local_TP_Name`** call to specify additional names to be matched with incoming **`Allocate`** requests.

This parameter is optional; if it is not specified, the default is **`CPIC_DEFAULT_TPNAME`** for an invoking application, and **`CPIC_DEFAULT_TP_NAME`** for an invoked application.

CSV_data

This section of the file specifies options for applications that use the CSV interface. It applies only to applications that use the CONVERT verb to perform character conversion with a user-defined conversion table (Type G); if no applications on the client use this function, there is no need to include this section. For more information on the CONVERT verb and a Type G conversion table, see Chpater 2 in the current *HP-UX SNAplus2 API CSV Programmer's Guide*. There is only one option in this section.

CSVTBLG The full pathname of the file containing the user-defined Type G conversion table. This parameter is required if CSV applications need to perform Type G character conversion (there is no default); otherwise it is optional.

LAN Access Timeout

If the client is communicating with SNAplus2 servers across a network for which connection charges are payable, you may want to ensure that the TCP/IP connection from the client is dropped automatically after applications on the client have stopped using SNAplus2 resources. Note that this does not automatically stop the NAP on the client; it remains active, and will attempt to re-establish contact with a server if an application requires it at a later time.

The **lan_access_timeout** parameter (described on page 22) allows you to do this. The TCP/IP connection will be dropped when there have been *none* of the following on the client for the specified time:

- APPC or CPI-C conversations active (or attempts to start a conversation)
- 3270 or LUA sessions enabled
- CSV TRANSFER_MS_DATA verbs
- Administration commands (except as noted below).

In particular, the TCP/IP connection will be dropped if you start the NAP but then do not start any SNAplus2 applications on the client within the specified timeout.

When one of the above events occurs while the TCP/IP connection is down, the client will restart the attempt to contact a server, as described for the [Servers] section of the file (beginning on page 22). Note that the following events will not cause the client to restart the connection:

- Error or audit messages logged by the client (these will be logged locally on the client, even if central logging is being used)
- The administration commands **query_central_logger** or **query_node_all** (these will return the information that was available before the TCP/IP connection was dropped, and so may not match the current status of the LAN).

Note also that incoming Attaches for invoked TPs on this client cannot be accepted while the TCP/IP connection is down; the Attach will be rejected as though the target system is inactive. This means that automatically-started TPs on the client will not be available if no other applications on the client are running and the TCP/IP connection has timed out. However, operator-started TPs on the client can be used at any time, because a **Receive_Allocate** issued by the TP will re-establish the TCP/IP connection.

Windows Client Invocable TP Data File: sna_tps.ini

The Windows Client invocable TP data file, **`sna_tps.ini`**, contains information that SNAplus2 needs when starting an invocable TP on a Windows Client—that is, an APPC TP that issues RECEIVE_ALLOCATE, or a CPI-C application that issues **`Accept_Conversation`**. In this section, the phrase “Receive_Allocate” is used to indicate either of these two API calls.

The TP can be started either by an operator or automatically by SNAplus2. The following table shows the difference:

How Started:	sna_tps.ini File Required	TP-Entry Required
Automatically	Yes	Yes
Operator	No[a]	No[a]

a. Default parameters are used.

The **`sna_tps.ini`** file is stored in the directory where the Windows software was installed (typically c:\windows); it is an ASCII text file that you create and modify using any standard Windows text editor. If you create a file and have a TP-entry in it, for both automatic and operator started TPs, one of the lines in the entry **must** be USERID (refer to page 34).

If the TP will be operator-started (not started automatically by SNAplus2), there is generally no requirement to set up any additional information. This section applies only in the following cases:

- The TP will be operator-started (not started automatically by SNAplus2), and will be a broadcast queued TP. In this case, the only information you need to provide is the TP name and the TP type QUEUED-BROADCAST.

- The TP will be operator-started (not started automatically by SNAplus2), and you need to specify the timeout value for a Receive_Allocate issued by the TP. In this case, the only information you need to provide is the TP name and the required timeout value. The TP type specified in the file must not be set to NON-QUEUED; it may be left as the default (QUEUED), or set to QUEUED-BROADCAST as above. (If the TP is not defined, the Receive_Allocate timeout is infinite; that is, the TP waits indefinitely for an incoming Allocate and does not time out.)
- The TP is to be started automatically by SNAplus2 when another TP requests a conversation with it.

NOTE:

If you need to restrict the TP to use particular options for conversation security, confirm synchronization, or conversation type (mapped or basic), or if you want to restrict the number of instances of the TP that can be running at any time, you also need to define the TP as a domain resource using the **`define_tp`** command. For more information, refer to the description of this command in Chapter 5 of the *HP-UX SNAplus2 Administration Guide*. In addition, if you need to set up environment variables for use with the TP, you should do this using an application-specific section of the win.ini file as usual.

SNAplus2 uses this file for the following purposes:

- When a TP issues Receive_Allocate, SNAplus2 searches this file for an entry with the appropriate TP name. If the entry exists, and includes a value for the Receive_Allocate timeout, SNAplus2 uses this value when processing the Receive_Allocate; otherwise it uses the default (no timeout - wait indefinitely).
- When an incoming Allocate request arrives at the target system, and the requested TP is not already running with a Receive_Allocate outstanding, SNAplus2 searches this file for an entry with the TP name specified on the incoming Allocate. If the entry exists, SNAplus2 uses the information in this entry to start the TP (if it is defined to be non-queued or is not already running), or to determine that it should queue the incoming Allocate (if the TP is already running and is defined to be queued).

sna_tps.ini File Format

Each entry in the file has the following format:

```
[TPname]

USERID = userid
PATH = full pathname of executable file
ARGUMENTS =  command-line arguments required by TP,
separated by spaces
TYPE =  QUEUED | QUEUED-BROADCAST | NON-QUEUED
TIMEOUT =  nnn
SHOW =  MAXIMIZED | MINIMIZED | HIDDEN | NORMAL |
NOACTIVATE | MINNOACTIVATE
```

NOTE: For information on USERID, refer to the description of this entry in Chapter 6 of the *HP-UX SNAplus2 Administration Guide*.

Note the following points about the format of these entries:

- You can include a comment line by including "#" as the first character of the line; SNAplus2 then ignores this line. SNAplus2 also ignores completely blank lines.
- Each "parameter = value" entry must be on one line; it cannot contain line-break characters. The maximum length of a line is 255 characters; additional characters are ignored.
- White space (space characters and tab characters) at the start or end of a line, or before or after the " = " character, is ignored.·
- Each TP definition begins with the line identifying the TP name, and ends with the end of the file or the next TP name.
- Do not attempt to specify the same keyword more than once for the same TP. Only the last instance of each keyword will be used.

sna_tps.ini File Contents

The parameters are as follows. For an operator-started TP, the only parameters used are the TP name, the TP type, and the timeout value; the other parameters apply only to automatically-started TPs.

TPname

The name of the TP (1 - 64 characters, with no embedded space characters). The TP name specified on the Receive_Allocate, or on the incoming Allocate request, is matched against this name. If the TP is an automatically-started TP, it must specify this TP name on the RECEIVE_ALLOCATE verb when it starts up, to allow SNAplus2 to route the incoming Attach to the correct TP.

This name is normally specified as an ASCII string, enclosed in double quotation marks, for example "TPNAME1". Alternatively, it may be specified as a hexadecimal array representing the EBCDIC characters of the TP name, for example <53504E414D45F1>; or as a combination of the two, for example <3f>"TP1" (the first character is the unprintable character 0x3f, and the following characters are "TP1").

SNAplus2 converts a supplied ASCII string to EBCDIC, but does not perform any conversion on a hexadecimal string (which is assumed to be already in EBCDIC). It then pads the name on the right with EBCDIC spaces to 64 characters before matching against the specified TP name.

PATH

The path and filename of the executable file for this TP. If you specify a filename with no path, SNAplus2 uses the normal Windows mechanisms for locating the executable file.

This line is optional; if it is not included, SNAplus2 assumes that the executable file name is the same as the TP name, and uses the normal Windows mechanisms for locating the executable file.

ARGUMENTS

Any command-line arguments to be passed to the TP, separated by spaces. These will be passed to the TP in the same order as they appear here.

This line is optional; if it is not included, the TP is invoked without any command-line arguments.

TYPE

Queued

Specify QUEUED if the TP is a queued TP; this indicates that any incoming Allocate requests arriving while the TP is running are queued until the TP issues another Receive_Allocate, or until it finishes running and can be restarted. An incoming Allocate request will be routed to this TP only if it is received by an LU that is configured to route incoming Allocate requests to this computer.

Queued-Broadcast

Specify QUEUED-BROADCAST if the TP is a broadcast queued TP; this indicates that any incoming Allocate requests arriving while the TP is running are queued until the TP issues another Receive_Allocate, or until it finishes running and can be restarted. When the TP is started, information about the TP is broadcast to all servers on the LAN; this means that, if an LU on another computer receives an incoming Allocate request and has no routing information configured, it can dynamically locate the TP and route the Allocate request to it.

Using QUEUED-BROADCAST instead of QUEUED avoids having to configure explicit routing information on LUs, and allows load-balancing by running more than one copy of the same TP on different computers. However, if you want to avoid broadcasting information in order to reduce LAN traffic, or if you need to ensure that incoming Allocate requests arriving at a particular LU are always routed to the same copy of the TP, you should use QUEUED.

Non-Queued

Specify NON-QUEUED if the TP is a non-queued TP; this indicates that SNAplus2 should start a new copy of the TP each time an incoming Allocate request arrives for it. Do not specify the TIMEOUT parameter for a non-queued TP.

NOTE: A TP defined as non-queued cannot be started by an operator; it is always started automatically by SNAplus2. Do not specify NON-QUEUED if the TP will be operator-started. If a user attempts to start a non-queued TP, SNAplus2 rejects the RECEIVE_ALLOCATE verb because there is no incoming Allocate request waiting for it.

If you use NON-QUEUED, there may be more than one copy of the TP running at a time. If the TP writes to files on the Windows computer, you need to ensure that different copies of the TP do not overwrite each other's files. To do this, use one of the following methods:

- Ensure that the TP appends data to an existing file instead of creating the file (so that all copies of the TP append data to the same file)
- Design the TP to generate filenames at run-time, based on the process ID with which the TP is running (so that each copy of the TP writes to a different file).

This line is optional; if it is not included, or if an invalid value is specified, the default is QUEUED.

TIMEOUT

The maximum length of time, in seconds, that a Receive_Allocate call issued by the TP should block if there is no incoming Allocate request pending. If no incoming Allocate is received in this time, the call will fail with a return code indicating `State check - Allocate not pending'.

A zero timeout value indicates that the call will always fail unless an incoming Allocate is already pending when the call is issued; a timeout value of -1 indicates that the call will wait indefinitely for an incoming Allocate and will not time out.

This line is optional; if it is not included, or if an invalid value (a non-numeric value) is specified, the default is -1 (infinite).

Do not specify this parameter if the TYPE parameter above is set to NON-QUEUED. SNAplus2 uses a zero timeout value for non-queued TPs, because the TP is always started in response to an incoming Allocate and so there will always be one pending.

SHOW

Specify how the application should be displayed when it is started. Note that this parameter is passed to the application, and not processed by SNAplus2; it is the application's responsibility to interpret it and act on it. Allowed values:

- MAXIMIZED The application is maximized.
- MINIMIZED The application is minimized.
- HIDDEN The application does not appear on the screen.
- NORMAL The application is displayed at its normal size and position.
- NOACTIVATE The application is displayed at its normal size and position, and the focus remains on the previously active window; this application's window does not become the active window.
- MINNOACTIVATE The application is minimized, and the focus remains on the previously active window.

This parameter is optional; if it is not included, the default is NORMAL.

3

TN Server Management and Configuration

This chapter tells you how to manage and configure TN3270 Servers.

TN Server Management and Configuration

Introduction

The SNAplus2 TN Server provides access to 3270 host computers for TN3270 users on other computers.

NOTE: In SNAplus2 manuals, the term "TN Server user" refers to the computer where a TN3270 program is running (identified by its TCP/IP address), and not to an individual user of that program.

The TN Server feature allows TN3270 users to share a host connection with SNAplus2 or with other TN3270 users, instead of requiring a direct link. It also allows TN3270 users to access hosts that are not running TCP/IP.

The SNAplus2 TN Server feature provides an association between a TN3270 user and a SNAplus2 3270 LU. All data from the TN3270 user is routed to the LU. This means that the configuration for both the host and the TN3270 user is as though they were connected directly; neither needs to be aware that data is being routed through TN Server.

TN Server Users

3270 emulation programs that communicate over TCP/IP rather than over an SNA network are referred to as TN3270 programs (Telnet 3270 emulation programs). SNAplus2 TN Server supports various TN3270 programs, including SNAplus2 TN3270 (packaged separately from the main SNAplus2 product).

When a TN3270 program communicates with TN Server, SNAplus2 identifies it by the TCP/IP address of the computer where the TN3270 program is running; it cannot distinguish between two different TN3270 programs being used by different users on the same computer.

NOTE: Each TN Server user is normally configured to access a single 3270 LU, and so is restricted to one host session at a time. However, you can also configure a TN Server user to access a pool of 3270 LUs, instead of having a single dedicated 3270 LU for each user. This allows the user to access as many sessions as there are available LUs in the pool.

Managing TN Server (Overview)

If TN3270 users will be using the TN Server feature on an SNAplus2 node to communicate with host systems, you need to define the communications link to the host.

The definition of the Link Station (LS) to the host must include the name of a local PU to own the 3270 LUs, and must have "**`solicit_SSCP_sessions`**" set.

You then need to define LUs that can be used for 3270 emulation, and optionally group these LUs into LU pools.

For more information, refer to "Administering specific SNAplus2 functions" in chapter 1 of the current *HP-UX SNAplus2 Administration Guide.*

To define the TN3270 users that can access TN Server, and assign them to SNAplus2 3270 LUs, use the following command:

`define_tn3270_access (Managing TN Server)`

To obtain information on the configuration of TN3270 users, use the following command:

`query_tn3270_access_def (Managing TN Server)`

To delete TN3270 users so that they can no longer use TN Server for 3270 emulation, use the following command:

`delete_tn3270_access (Managing TN Server)`

define_tn3270_access

`define_tn3270_access` identifies a TN3270 user on another computer that can use the TN Server feature of SNAplus2 to access a host for 3270 emulation, and defines the 3270 LUs available to that user. This access facility can be used to:

- define a new user
- define new sessions for use by an existing user
- modify the session parameters for an existing user.

To delete sessions from an existing user, use **`delete_tn3270_access`**.

Supplied Parameters

Table 1

Parameter Name	Type	Length	Default
`Define_tn3270_access`:			
default_record	constant		NO
client_address	character	64	
description	character	31	(null string)
address_format	constant		
`tn3270_session_data`:			
description	character	31	(null string)
port_number	decimal		
lu_name	character	8	
session_type	constant		3270_Display_Model_2
model_override	constant		YES

One or more **`tn3270_session_data`** subrecords can be included.

Command Parameter Descriptions

The following parameters are used to define the tn3270 access facility:

default_record Specifies whether this command defines the default record, which will be used by any TN3270 user not explicitly identified by a TCP/IP address. If a TN3270 user attempts to contact the TN Server node, and the user's TCP/IP address does not match any **`define_tn3270_access`** record in the configuration but there is a default record defined, the parameters from this record will be used. Allowed values:

YES	This command defines the default record. Do not specify the client_address and address_format parameters.
NO	This command defines a normal TN3270 user record.

NOTE:

A default record provides access to the TN Server function for any TN3270 user that can determine the TCP/IP address of the computer where the TN Server is running. To restrict the use of TN Server to a specific group of users, either do not include the default record, or leave it with no 3270 LU or LU pool configured so that it cannot be used.

You can also set up a default record for most users, but explicitly exclude one or more TCP/IP addresses. To do this, define the addresses to be excluded as TN Server users, and leave them with no 3270 LU or LU pool configured.

client_address The TCP/IP address of the computer on which the TN3270 program runs. This can be specified as a dotted-decimal IP address (such as **`193.1.11.100`**), as a name (such as **`newbox.this.co.uk)`**, or as an alias (such as **`newbox`**). If you use a name or alias, the following restrictions apply:

- The UNIX computer must be able to resolve the name or alias to a fully-qualified name (either using the local TCP/IP configuration or using a Domain Name Server).
- Each name or alias must expand to a unique fully-qualified name; you should not configure two names for users of the same TN Server node that will be resolved to the same fully-qualified name

description An optional string of 0 - 31 characters. The string is for information only; it is stored in the configuration file and returned on **`query_tn3270_access_def`**, but SNAplus2 does not use it. You can use it to store additional information to help distinguish between users.

address_format Specifies the format of the **client_address** parameter. Allowed values:

IP_ADDRESS	IP address
FULLY_QUALIFIED_NAME	Alias or fully-qualified name

Each TN3270 user can access the same TN Server node with multiple sessions, by using a different TCP/IP port for each session. For each of these sessions, include a **tn3270_session_data** subrecord with the following information:

Subsection Parameter Descriptions

description An optional string of 0 - 31 characters. The string is for information only;
it is stored in the configuration file and returned on **query_tn3270_access_def**, but SNAplus2 does not use it.

port_number The number of the TCP/IP port that the TN3270 program uses to access the TN Server node. If the port number matches an existing port number defined for one of this TN3270 user's sessions, the information for that session is replaced; otherwise a new session is added.

If the TN3270 program uses TCP/IP port number 23 (the port number used by the Telnet daemon program on the UNIX computer), you will need to set up an additional initialization file to share this port number between TN Server and the Telnet daemon program. For more information, refer to page 46 .

lu_name Name of the LU or LU pool that this session uses. This is a type A EBCDIC string (starting with a letter). It must match the name of a type 0 - 3 display LU defined on this node, or an LU pool containing LUs on this node.

If you specify an LU name, a TN3270 program with the specified TCP/IP address and TCP/IP port number will be able to use only one session at a time through this TN Server node. If you specify an LU pool, the program can use multiple sessions (or multiple copies of the program can access sessions using this TN Server), up to the number of LUs on this node that are available from the pool.

session_type Screen model of the LU or pool. Allowed values:

`3270_DISPLAY_MODEL_2`
Display model 2 (80 x 24)

`3270_DISPLAY_MODEL_3`
Display model 3 (80 x 32)

`3270_DISPLAY_MODEL_4`
Display model 4 (80 x 43)

`3270_DISPLAY_MODEL_5`
Display model 5 (132 x 27)

model_override Specifies whether the user has permission to change the session to use a different screen model from the one specified. Allowed values:

YES	The user can change the screen model.
NO	The user cannot change the screen model.

Error Return Codes

If the command cannot be executed, SNAplus2 returns the following parameters:

primary_rc	PARAMETER_CHECK
secondary_rc	INVALID_SESSION_TYPE The session_type parameter for one or more sessions was not set to a valid value. UNKNOWN_CLIENT_ADDRESS The specified name or alias could not be mapped to a fully-qualified name. CLIENT_ADDRESS_CLASH The fully-qualified name, resolved from the **`client_address`** parameter, clashes with one that has already been defined. TCPIP_PORT_IN_USE The TCP/IP port number cannot be used by TN Server

because it is already in use by a different program.

Information on Common Return Codes from **snapadmin** commands can be found in “Common Return Codes from snapadmin commands” in chapter 5 of the current *HP-UX SNAplus2 Administration Guide*.

For more information, refer to “Administering specific SNAplus2 functions” in chapter 1 of the current *HP-UX SNAplus2 Administration Guide*.

Using the Telnet Daemon's TCP/IP Port

If you are setting up TN Server for use with a TN3270 program that uses TCP/IP port number 23, you will need to set up the UNIX computer where the node owning this TN Server runs, to share this port number between TN Server and the Telnet daemon program. To do this, take the following steps:

1. Ensure that the SNAplus2 software is stopped on the UNIX computer (by using **snap stop**).
2. Log on to the UNIX computer as root.
3. Edit the file **/etc/inetd.conf**, and find the line beginning with `telnet'. Make a note of the executable name and any supplied parameters for the Telnet daemon program, which are included in this line; typically these are **/usr/lbin/telnetd** and **telnetd**. Comment out this line by inserting a # character at the start of the line, and save the file.[1]
4. Create an ASCII text file **/etc/snainetd.conf**. This file should consist of a single line containing the Telnet daemon executable name and parameters, as determined in step 3, for example:
   /usr/lbin/telnetd telnetd -b/etc/motd
5. Edit the file **/etc/rc_config.d/snaplus2**. Set the **START_SNAINETD** variable to the value 1 if the value is set to 0 (default). Because this is a shell variable, do not insert a blank or tab.

1. All continuation commented lines should start with the # character. For example, if a line continues to a second line, the second line should also contain a # charater at the start of the line.

6 Restart the SNAplus2 software, with the command:

```
/sbin/init.d/snaplus2 start
```

The snaplus2 shell script ensures the correct activation of **`inetd`** and **`snapinetd`**.

NOTE: Once **`/etc/rc_config.d/snaplus2`** is updated, there is no need to do anything at the startup of UNIX HP-UX.

query_tn3270_access_def

`query_tn3270_access_def` returns information on TN3270 users on other computers that can use the TN Server feature of SNAplus2 to access a host for 3270 emulation. It can return either summary or detailed information, about a single user or multiple users, depending on the options used.

Supplied Parameters

Table 2

Parameter Name	Type	Length	Default
query_tn3270_access_def			
num_entries	decimal		1
list_options	constant		LIST INCLUSIVE
default_record	constant		NO
client_address	character	64	(null string)
port_number	decimal		(none specified)

num_entries

Maximum number of users for which data should be returned. If detailed information on user sessions is being returned, note that this number includes partial entries (for which a client address is specified, so that the returned data does not include the user definition or the user's first session).

To request data for a specific user rather than a range, specify the value 1. To return all entries, specify zero.

list_options

The position in the list from which SNAplus2 should begin to return data, and the level of information required for each entry. Specify the level of information with one of the following values:

SUMMARY Summary information only.

DETAIL Detailed information. Combine this value using a " + " character with one of the following values:

FIRST_IN_LIST
Start at the first session for the first user in the list.

LIST_INCLUSIVE
Start at the entry specified by the supplied client address and port number, or start at the first session for the specified client address if no port number is specified.

LIST_FROM_NEXT
If a port number is specified, start at the session immediately following the session with the specified port number. If no port number is specified, start at the first session for the specified client address.

default_record

Specifies whether the requested entry (or the entry to be used as an index into the list) is the default record.

To query the default record, which is used by any TN3270 user not explicitly identified by a TCP/IP address, specify YES, and do not specify the **`client_address`** parameter.

To query a normal TN3270 user record, specify NO.

client_address

The TCP/IP address of the TN3270 user for whom information is required, or the name to be used as an index into the list of users. This parameter is ignored if **`list_options`** is set to FIRST_IN_LIST.

The address can be specified as a dotted-decimal IP address (such as 193.1.11.100), as a fully-qualified name (such as newbox.this.co.uk), or as an alias (such as newbox).

address_format

Specifies the format of the **`client_address`** parameter. Allowed values:

IP_ADDRESS IP address

FULLY_QUALIFIED_NAME Alias or fully-qualified name

port_number

To return information starting with a specific session for the specified client address, set this parameter to the TCP/IP port number defined for that session. To return information starting at the first session for the specified client address, do not specify this parameter.

Returned Parameters - Summary Information

Table 3

Parameter Name	Type	Length
default_record	constant	
client_address	character	64
address_format	constant	

default_record

Specifies whether this entry is the default record. Possible values:

YES This is the default record. The **`client_address`** parameter is not used.

NO This is a normal TN3270 user record.

client_address

The TCP/IP address of the TN3270 user.

address_format

Specifies the format of the client_address parameter. Possible values:

IP_ADDRESS IP address

FULLY_QUALIFIED_NAME Alias or fully-qualified name

Returned Parameters - Detailed Information

Table 4

Parameter Name	Type	Length
default_record	constant	
client_address	character	64
description	character	31
address_format	constant	
tn3270 session data:		
description	character	31
port_number	decimal	
lu_name	character	8
session_type	constant	
model_override	constant	

default_record

Secifies whether this entry is the default record. Possible values:

YES This is the default record. The **`client_address`** parameter is not used.

NO This is a normal TN3270 user record.

client_address

The TCP/IP address of the TN3270 user.

address_format

Specifies the format of the **`client_address`** parameter. Possible values:

IP_ADDRESS IP address

FULLY_QUALIFIED_NAME Alias or fully-qualified name

For each session defined for the specified (defined by its TCP/IP port number), a **`tn3270_session_data`** subrecord is returned, containing the following parameters:

port_number

The number of the TCP/IP port that the TN3270 program uses to access the TN Server node.

description

An optional string describing the session.

lu_name

Name of the LU or LU pool that this session uses.

session_type

Screen model of the LU or pool. Possible values:

3270_DISPLAY_MODEL_2
Display model 2 (80 x 24)

3270_DISPLAY_MODEL_3
Display model 3 (80 x 32)

3270_DISPLAY_MODEL_4
Display model 4 (80 x 43)

3270_DISPLAY_MODEL_5
Display model 5 (132 x 27)

model_override

Specifies whether the user has permission to change the session to use a different screen model from the one specified. Possible values:

YES	The user can change the screen model.
NO	The user cannot change the screen model.

Error Return Codes

If the command cannot be executed, SNAplus2 returns the following parameters:

primary_rc PARAMETER_CHECK

secondary_rc One of the following:

INVALID_CLIENT_ADDRESS
The **`list_options`** parameter was set to LIST_INCLUSIVE, but the **`client_address`** parameter did not match the address of any defined TN3270 user.

INVALID_PORT_NUMBER
The **`list_options`** parameter was set to LIST_INCLUSIVE, but the **`port_number`** parameter did not match a port number defined for the specified TN3270 user.

INVALID_LIST_OPTION
The **`list_options`** parameter was not set to a valid value.

Information on Common Return Codes from **`snapadmin`** commands can be found in “Common Return Codes from snapadmin commands” in chapter 5 of the current *HP-UX SNAplus2 Administration Guide*.

delete_tn3270_access

`delete_tn3270_access` is used to do one of the following:

- Delete a TN3270 user, so that this user can no longer use TN Server to access a host
- Delete one or more of the user's sessions but leave the user configured.

Supplied Parameters

Table 5

Parameter Name	Type	Length	Default
Delete tn3270 access:			
default_record	constant		No
client_address	character	64	
delete_options	constant		
port_number	decimal		

If **`delete_options`** is not specified, one or more **`port_number`** parameters can be included.

default_record

Specifies whether this command refers to the default TN3270 user record that is used by any TN3270 user not explicitly identified by a TCP/IP address (deleting this record means that such users cannot access TN Server). Allowed values:

YES — This command refers to the default TN3270 user record. The **`client_address`** parameter is not used.

NO — This command refers to a normal TN3270 user record.

client_address

The TCP/IP address of the TN3270 user to be deleted. This can be specified as a dotted-decimal IP address (such as 193.1.11.100), as a fully-qualified name (such as newbox.this.co.uk), or as an alias (such as newbox), as specified on the **`define_tn3270_access`** command.

delete_options

To delete one or more sessions but leave other sessions configured, do not specify this parameter, and specify the sessions to be deleted using **`port_number`** parameters. Otherwise, specify one of the following values:

ALL_SESSIONS
Delete all sessions but leave the TN3270 user configured.

DELETE_USER
Delete the user and all the user's sessions.

If the delete_options parameter is not specified, specify each session to be deleted as follows (using port_number):

port_number

The TCP/IP port number used for the session.

Error Return Codes

If the command cannot be executed, SNAplus2 returns the following parameters.

primary_rc PARAMETER_CHECK

secondary_rc One of the following:

INVALID_CLIENT_ADDRESS
The specified client address did not match the TCP/IP address defined for any TN3270 user.

INVALID_PORT_NUMBER
The specified TCP/IP port number did not match any TCP/IP port number defined for this user.

Information on Common Return Codes from **snapadmin** commands can be found in “Common Return Codes from snapadmin commands” in chapter 5 of the current *HP-UX SNAplus2 Administration Guide*.

4

Changes to SNAPADMIN

This chapter describes changes to the **snapadmin** command.

Command Line Help

SNAplus2 R5.2 provides command-line administration on-line help that is consistent and easy to use, and can provide you with either an overview or a specific set of detailed information.

The following commands are provided:

snapadmin	Usage statement.
snapadmin -h	Basic help, and help on help.
snapadmin -h -d	Detailed help (that lists available commands).
snapadmin -h <command>	Help on named command.
snapadmin -h -d <command>	Detailed help on named command (lists fields)

The **-d** option always provides detailed information which expands on the help provided without this option.

Usage Statement

The usage statement now includes all the available command-line options.

snapadmin

Usage:

snapadmin [-n node] [-d] [-a] [-h] <-i infile> | <command>

-n node	Send command to named node in the same SNA domain. By default, node commands are sent to the local node.
-i infile	Use commands from named file
-d	List detailed information (query commands only)
-a	List all items (query commands only)
-h	Display help
-c	Change specified parameters on an existing item. See page 67 .

General Help

The on-line help describes how to enter information into fields, and how to obtain information on the available commands.

The detail option -d lists all the available commands.

Normal Help

`snapadmin -h`

Using snapadmin commands:

A command consists of the command name, followed by a "parameter=value" entry for each parameter you wish to specify, each separated by commas. Some commands may also take subrecords which are included in braces as follows:

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

Parameters may take values of the following types:

Examples	
Character String	ABC
Decimal number	12
Hex number preceded by 0x	0x0A
Hex array of defined length preceded by 0x or enclosed in <>	0x100010D967AB <100010D967AB>
Defined constant	YES

Detailed Help

`snapadmin -h -d`

This option lists all the available commands.

The list of commands is formatted to fit the width of the screen . (No attempt is made to limit the size of the output to the number of lines on the screen.)

Help On Command

This provides you with information about a particular command.

Normal Help on Command

This command gives a brief description of the command's function. For example:

```
snapadmin -h define_lu_0_to_3
```

Defines an LU for use with 3270 emulation or LUA (an LU of type 0, 1, 2 or 3) and optionally assigns the LU to an LU pool.

Detailed Help on Command

Using the detail option **`-d`** lists both mandatory and optional fields for a command, and describes the type and range of each, as well as displaying the brief functional description.

Lists of constants are formatted to fit into the width of the screen. For example:

```
snapadmin -h -d define_lu_0_to_3
```

Defines an LU for use with 3270 emulation or LUA (an LU of type 0, 1, 2 or 3) and optionally assigns the LU to an LU pool.

Name	Type	Length
Mandatory fields:		
lu_name	character	8
nau_name	decimal	1-255
pu_name	character	8
Optional fields:		
pool_name	character	8
priority	constant	NETWORK, HIGH, MEDIUM, LOW
lu_model	constant	3270_DISPLAY_MODEL_2, 3270_DISPLAY_MODEL_3, PRINTER, RJE_WKSTN, UNKNOWN
description	character	31

Additional Status Commands

The following new commands provide you with concise summaries of commonly-required status information. The information provided is similar to the contents of the node and domain windows of the Motif administration program.

snapadmin status_connectivity	DLCs, ports and LSs
snapadmin status_dependent_lu [,pu_name=pu_name,lu_type=lu_type]	Dependent LUs type 0-3 and 6.2 and their owning PUs
snapadmin status_dlur [,pu_name=dlur_pu_name]	DLUR PUs and LUs
snapadmin status_lu62	Dependent and independent LUs type 6.2
snapadmin status_node	All nodes in the domain.
snapadmin status_all [,node_name=node_name]	All the above arranged by node.

All of these commands, with the exception of status_node, can be sent to a particular node using the **-n** node parameter. All nodes must be in the same SNA domain. By default they go to the local node.

The description strings (truncated if necessary to fit the width of the display) for each resource are displayed if there is enough room.

Connectivity Status

```
>snapadmin status_connectivity
```

This command shows the status of all the DLCs, ports and link stations on the node.

Each resource is shown as being in one of the following states:

- Inactive
- Active
- Starting
- Stopping
- On demand (link stations only)
- Disabled (link stations only)

Dependent LU Status

`>snapadmin status_dependent_lu`

This command shows the status of all the dependent LUs on the node.

The following status information is available:

❒ PUs are displayed as either "Inactive" or "SSCP" depending on whether the PU-SSCP session is active.

❒ Each LU on the PU is displayed as:

- "Inactive" if the LU-SSCP session is inactive
- "SSCP" if the PLU_SLU session is inactive
- "Active" if neither of the above apply

Additional information is displayed for an LU if it is in use by an application. The information varies according to the type of the application.

Application Type	Information Displayed
Unknown or LUA application	Unknown/LUA application
FMI application - 3270	3270 display user: user_name Computer: system_name or 3270 printer user: user_name Computer: system_name
FMI application - RJE	RJE workstation: wkstn_name Computer: system_name

Application Type	Information Displayed
TN3270 applications	If the configured address format is a dotted decimal: TN3270 address: 1.2.3.4 otherwise: TN3270 address: 1.2.3.4 (host_name)
Dependent LU6.2	Partner LU: fqplu_name Mode: mode_name

Status about a particular PU

Status about a particular PU is obtained by specifying the PU in the command, for example:

```
>snapadmin status_dependent_lu,pu_name=ETH0
```

Status about a particular LU type

Status about a particular LU type is obtained by specifying the LU type in the command. The following values are valid:

- DISPLAY: 3270 display LU
- PRINTER: 3270 printer LU
- RJE: RJE LU
- LU62: LU6.2 dependent LU
- OTHER: Unrestricted type.

For example:

```
>snapadmin status_dependent_lu,lu_type=DISPLAY
```

Note that:

- The Motif Administration program merges the concepts of LS and PU for simplicity (except where there are DLUR PUs), but the command-line administration program does not.
- Status information about 3270 users or TN3270 clients is displayed by the command-line administration program as part of the dependent LU status information.

LU6.2 Status

>snapadmin status_lu62

This command displays the status of dependent and independent LUs type 6.2, giving a current session count for each LU-PLU-mode triplet that is currently active or has been active since the node was started. The machine field displays the target computer for any incoming attaches (that is, the computer where the TP is running).

NOTE: The Motif administration program shows information about dependent and independent LUs separately (in different panes).

DLUR Status

>snapadmin status_dlur

This command displays the status of PUs using DLUR on the node and their LUs.

The DLUS with which a PU or LU has an SSCP session established appears under the DLUS column. This is marked "Inactive" if no SSCP session is active.

If an LU has a PLU-SLU session active, the PLU name is displayed. This is also marked "Inactive" when there is no active session

For dependent LU status, the status of a particular PU can be obtained by specifying the PU name

Node Status

>snapadmin status_node

This command lists the nodes in the domain and gives their status, configuration role and description. Status is one of the following:

- Inactive
- Active
- Starting
- Stopping

The configuration role is marked as either "Master" or "Backup", or left blank.

All Status

>**snapadmin status_all**

This combines the information from all the status queries into one command.

If a node does not support DLUR, no DLUR status is shown. If there are no items defined for a particular status type, that section does not appear, so for example a node with no PUs does not have a dependent LU section.

Status for a particular node can be obtained by specifying the node name in a parameter node_name, otherwise information about all nodes in the domain is produced.

Other Command-Line Usage Enhancements

In addition to on-line help, the command-line administration provides the following additional enhancements in Release 5.2:

- Queries now list all items by default (so that there is no need to specify the -a option).
- Hex field entry has been simplified.
- Individual parameters can be overwritten on some defines.

This section describes these enhancements in more detail.

Queries List all Items by Default

Command-line administration query commands previously expected the name of a resource to query (for example, snapadmin query_ls expected a link station to be named). To list all the resources of a particular type (for example, all the defined link stations on the node) the user had to explicitly enter the -a option.

The -a option is now implied by default if no particular item is named.

For back compatibility the -a option is still supported.

Hex Field Entry

Hex fields previously had to be enclosed in angle brackets.

Hex fields can now be preceded by the characters 0x instead. The value entered must still be of the correct length.

For back-compatibility the angle-bracket format will continue to be accepted, and fields will continue to be written to the configuration files in this format.

Overwrite Individual Parameters

Previously the entire command to change a particular parameter on an existing item had to be re-entered.

A new command-line option, **`-c`** allows the user to change only the specified parameters on an existing item just by specifying the item name and the parameters to be changed.

This operation can only be used to change information that can currently already be changed using a **`define_*`** command.

For example, it is not possible to change the NAU address of an existing LU type 0-3 using a **`define_lu_0_to_3`** command. Instead the LU needs to be deleted and then re-defined. In the same way, neither is it possible to change the NAU address of an existing LU type 0-3 using the **`-c`** option.

This function is supported on the following verbs:

define_node
define_sdlc_dlc
define_sdlc_port
define_sdlc_ls
define_qllc_dlc
define_qllc_port
define_qllc_ls
define_ethernet_dlc
define_ethernet_port
define_ethernet_ls
define_tr_dlc
define_tr_port
define_tr_ls
define_lu_0_to_3
define_rje_wkstn
define_local_lu
define_partner_lu
define_mode
define_tp
define_userid_password
define_cpic_side_info
define_downstream_lu

The **`-c`** option is ignored if it is used on any other verbs.

5

Additional Changes

This chapter describes changes to APPC, CPI-C, LUA, NOF, MS, CSV, and HLLAPI.

New Modes of Operation

SNAplus2 R5.2 introduced two new modes of operation, while continuing to support existing applications without change. Developers of new applications, or developers simply re-linking existing applications should read this material to ensure that the most appropriate options are chosen.

There are now three different modes of operation. Each is described in detail below.

- Back-compatibility mode continues to support all applications from previous SNAplus2 releases. It is suitable for old applications (which can run unchanged), or for applications which cannot be linked with either the POSIX or DCE thread libraries. This mode has a number of restrictions which the following two new modes address. We therefore recommend that new applications do not use this mode.
- Standard mode is a new mode that provides support for applications which are linked with either the POSIX or DCE thread library. The applications themselves can be either single or multi-threaded. Most newly developed applications should use this mode.
- Application scheduled mode is another new mode that provides the application increased control over the scheduling of SNA callbacks. This requires additional code in the application, but is particularly useful for applications which use MOTIF, or which have main scheduling loops which use the select() or poll() system call.

The sections below describe each mode in detail, followed by guidance on how to select which mode to use for a particular application.

Back-Compatibility Mode

Back-compatibility mode provides binary compatibility for existing applications, and is also suitable for use with new applications which are unable to link with either the POSIX or DCE thread libraries.

It is identical to the API support provided by the libmgr.sl library in SNAplus2 R5.0 and R5.1. In this mode the SNA API libraries make use of the old UNIX V.3 signal calls (sigset, sighold, sigrlse, sigpause, and so forth) and require the application to link with the libV3.sl library to access them.

Synchronous verbs are implemented by the library sleeping in poll(). Asynchronous verbs are implemented by making callbacks to the application from signal catcher context.

There are a number of disadvantages with this mode of operation compared to the new modes provided in this release.

- It is difficult to write an application that receives work from multiple sources. The only option is to use signals to do this.
- Applications are locked into using UNIX V.3 signal calls as these cannot be mixed with BSD or POSIX signal calls in the one application.
- Only a subset of system calls can be made from signal catcher context (and therefore from API callback context).
- It is not thread-safe.

Because of these disadvantages, we recommend that new applications do not use this mode.

Standard Mode

Standard mode allows SNA applications to take advantage of either POSIX or DCE threads. It is fully thread safe and avoids the disadvantages listed above for back-compatibility mode.

This mode is suitable for single-threaded as well as multi-threaded applications and likely to be the most appropriate mode to use with newly developed applications. In many cases it is easy to modify an old application using back-compatibility mode to use standard mode. For example, if the application does not use signal calls, a simple re-link may be all that is needed to switch over to using standard mode.

To use this mode, link the application with the new libsna.sl library (in place of the old libmgr.sl or libmgrdce.sl). Also, link the application with either libdce.sl or libpthread.sl to provide the thread support—even if the application is only single-threaded.

Standard mode makes no use of signals (no link required to libV3.sl). An additional thread is spawned within the library to listen for all SNA events. Asynchronous API calls are implemented by the library making callbacks to the application from the SNA library's listening thread.

This mode is compatible with the old DCE thread support that was provided with the libmgrdce.sl library. Applications can either be re-linked to use the new libsna.sl library, or use the old libmgrdce.sl library (which is implemented as a symbolic link to libsna.sl).

Application Scheduled Mode

Application scheduled mode is provided for applications that need increased control over the scheduling of events from multiple work sources. This is achieved by the SNA library not making API callbacks asynchronously (from a signal catcher in back-compatibility mode, or from a separate thread in standard mode), but by providing an extra SNA event handler entry point into the SNA library that the application can call. Callbacks are then made from within the context of this SNA event handler, or from within other calls into the SNA library.

For this to be useful, a method is needed to notify the application when SNA events arrive, and therefore that the application needs to call the SNA event handler to process events. This is achieved by giving applications access to the file descriptor over which SNA events arrive.

Applications require additional code to provide a mechanism for scheduling API callbacks and therefore need to be designed to support it. Modifying an existing application to use this mode may not require substantial changes as only the scheduling of events needs to be changed. All standard API calls (as documented in the various Programmer's Reference manuals) remain unchanged.

This mode is provided by the libsna.sl library. However, in this mode it is not necessary to link with the DCE or POSIX threads libraries as there is no need for the SNA library to spawn a listening thread as is done in standard mode. However, application scheduled mode is fully thread-safe, and the application can use either DCE or POSIX threads if necessary.

There are two variants of the application scheduled mode:

- One is suitable for applications that use the MOTIF or Xt libraries and use the scheduling techniques provided by the Xt library.
- The other is suitable for applications that use select() or poll() in their main scheduling loop to block for work from multiple sources.

Using Xt Scheduling

To use application scheduled mode in an application whose main scheduling loop is provided by the Xt library (for example, XtMainLoop), the following code must be added to your application before any other calls into any SNA library.

```
int app_context;
...
XtAppInitialize(&app_context...);
...
SNA_USE_XT_SCHED(app_context);
```

You should refer to Xt documentation before using **`XtApplnitialize`**.

The SNA_USE_XT_SCHED() call has no error return values.

The SNA_USE_XT_SCHED() works by calling XtAppAddInput() to register the SNA work sources. Whenever work arrives on the SNA file descriptor, the Xt library scheduling loop calls the SNA event handler directly, and this in turn makes any API callbacks to the application that may be required.

Using select() or poll()

To use application scheduled mode in an application whose main scheduling loop uses select() or poll(), the following call must be added to your application before any other calls into any SNA library.

```
SNA_USE_FD_SCHED();
```

This call has no error return values.

In addition, the main scheduling loop needs to be coded to obtain, and track any changes to, the SNA file descriptor. This file descriptor does not normally change, but does so, if for example, the SNA software is stopped. The following function is provided to report the SNA file descriptor to the application.

```
int SNA_GET_FD_();
```

The return value of this function is the file descriptor of the SNA work source. This will be either a valid file descriptor, or -1 indicating that there is no valid SNA file descriptor.

This call should be inserted into the main scheduling loop of the application, prior to the call to select() or poll(). It is important that there be no calls into any SNA libraries between the call to SNA_GET_FD() and the call to select() or poll().

The application must register for "read" events on the SNA file descriptor in the select() or poll() call. If it detects any of these events it must call the SNA event handler as follows.

```
SNA_EVENT_FD_();
```

This call has no error return values.

Choosing Which Mode to Use

If you have an existing application which works on SNAplus2 R5.0 or R5.1 you can continue to use it unchanged and without re-linking. The old libmgr.sl and libmgrdce.sl are still available and fully supported.

If you are making small changes to an application that uses back-compatibility mode, it is probably best to continue to use this mode. However, beware of the restrictions noted in section 3.2 above.

If you are writing a brand new application, or making significant changes to an existing one, we recommend you use either standard or application scheduled mode as these avoid the limitations of back-compatibility mode.

- If you are writing a MOTIF or Xt application, then application scheduled mode is the obvious choice.
- If you are writing a multi-threaded application then standard mode is the obvious choice.
- If you are writing a sophisticated application that processes work from multiple work sources using select()or poll(), then application scheduled mode is probably the best choice.
- Otherwise, standard mode is likely to be most appropriate.

Building Applications

Applications are compiled in exactly the same way for all three modes. The available verbs and the use of header files described in the appropriate Programmer's Reference remain unchanged.

The different modes are selected as follows.

- Back compatibility mode is obtained by linking with the following options, where <API> is the appropriate SNA API library

```
cc -L /opt/sna/lib -l<API> -lmgr
cc -L /opt/sna/lib -l<API> -lmgrdce -ldce
```

- When compiling code to run in standard mode the **-D_REENTRANT** option must be specified as it is required by the thread libraries. The application is then linked with one of the following options:

```
cc -L /opt/sna/lib -l<API> -lsna -ldce
cc -L /opt/sna/lib -l<API> -lsna -lpthread
```

- Application scheduled mode is obtained by linking with the following options, where <API> is the appropriate SNA API library

```
cc -L /opt/sna/lib -l<API> -lsna
```

Changes to HLLAPI

SNAplus2 R5.2 introduces additional options for HLLAPI applications. Existing applications will continue to operate without change. However, a new HLLAPI library is provided that supports additional features.

Back-Compatible HLLAPI Library

The old libhapi.sl library is retained unchanged. It has the following restrictions.

- It is not thread safe. Multi-threaded applications cannot use it.
- It uses the UNIX V.3 signal calls (sigset, sighold, sigrlse, sigpause etc.) and requires the application to link with the libV3.sl library to access them. Applications are then prevented from using other signal calls such as BSD or POSIX.

Existing applications can continue to use this library without change.

Thread-Save and POSIX HLLAPI Library

A new libhapip.sl library is provided that adds support for the following features.

- It is thread-safe. HLLAPI applications can now be multi-threaded, using either DCE or POSIX threads.
- It uses POSIX signal calls rather than the older UNIX V.3 calls. This makes it easier for the application to make its own signal calls without interfering with the operation of HLLAPI.

There are no changes to the verbs or the HLLAPI Programmer's Reference. To use the new version of HLLAPI, the application needs to be linked with the new library as follows.

```
cc -L /opt/sna/lib -lhapip
```