IBM Communications Server for Linux

Administration Command Reference

Version 6.2.3

SC31-6770-02

IBM Communications Server for Linux

Administration Command Reference

Version 6.2.3

SC31-6770-02

Note:
Before using this information and the product it supports, be sure to read the general information under Appendix D, "Notices," on page 603.

Fourth Edition (December 2007)

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

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

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

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

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

- Fax (USA and Canada):

  1-919-254-4028

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

  comsvrcf@us.ibm.com

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

Contents

Tables

About This Book

Communications Server for Linux Administration Command Reference contains information about starting and managing IBM® Communications Server for Linux® , an IBM software product that enables a computer running Linux to exchange information with other nodes on an SNA network.

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

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

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

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

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

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

Who Should Use This Book

This book is intended for system administrators who install Communications Server for Linux, configure the system for network connection, and maintain the system. They should be familiar with the hardware on which Communications Server for Linux operates and with the Linux operating system. They must also be knowledgeable about the network to which the system is connected and understand SNA concepts in general.

How to Use This Book

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

Organization of This Book

This book is organized as follows:

- Chapter 1, "Introduction," on page 1, provides an overview of the tasks involved in administering Communications Server for Linux, provides an overview of how to use the **snaadmin** administration program, and describes characteristics (such as parameter type) that are common to parameters used by all commands.
- Chapter 2, "Administration Commands," on page 11, provides detailed information about the parameters required for specific administration operations such as defining, starting, or querying a particular resource.

- Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists error return codes that are common to all commands.
- Appendix B, "Configuration Files," on page 593, describes the contents of the data files that control how Communications Server for Linux operates and explains how to modify these files if required.
- Appendix C, "Environment Variables," on page 599, provides a brief summary of all the environment variables that are used by Communications Server for Linux programs.

Typographic Conventions

Table 1 shows the typographic styles used in this document.

Table 1. Typographic Conventions

Special Element	Sample of Typography
Document title	*Communications Server for Linux Administration Command Reference*
File or path name	**sna.err**
Directory name	**/var/opt/ibm/sna**
Program or application	**snaadmin**
Command or utility	**define_node**; **snahelp**
General reference to all commands of a particular type	For example, **query_*** indicates all the administration records that query details of a Communications Server for Linux resource (*query_cn*, *query_cos*, *query_dlc*, and so on).
Option or flag	**-d**
Parameter	*lu_name*
Literal value or selection that the user can enter (including default values)	`255`
Constant (one of several possible parameter values)	`FIRST_IN_LIST`
Return value	`INVALID_LU_ALIAS`
Variable representing a supplied value	*infile*
Environment variable	`LD_RUN_PATH`
User input	**snaadmin status_dependent_lu,pu_name=ETH0**
Function, call, or entry point	`ctime()`
Data structure	`alert_3270_data`
Field name (in a data structure)	`c_cflag`
Keyboard keys	**ENTER**
Hexadecimal value	`0x20`

Related Publications

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

- *IBM Systems Network Architecture:*
  - *LU 6.2 Reference—Peer Protocols*, SC31-6808
  - *APPN Architecture Reference*, SC30-3422.
  - *Management Services*, SC30-3346
  - *Formats*, GA27-3136
  - *Technical Overview*, GC30-3073
- *IBM Communications Manager/2:*

  - *System Management Programming Reference*, SC31-6173
  - *APPC Programming Guide and Reference*, SC31-6160
- *IBM System/390 Principles of Operation*, SA22-7201
- *IBM z/Architect Principles of Operation*, SA22-7832

Chapter 1. Introduction

Communications Server for Linux administration commands are accessible through the **snaadmin** program. The **snaadmin** program is a command-line administration program that can be used to configure and manage the Communications Server for Linux node. The *Communications Server for Linux Administration Guide* describes how to configure and manage the Communications Server for Linux node using specific administration commands.

This book describes how to use the **snaadmin** program and the commands that you can issue through **snaadmin**. Administration commands are used for configuring, checking status of, and managing resources. Most administration commands belong to one of these categories as follows:

Configuring
The following types of commands are used to configure resources:

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

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

Checking status
The following types of commands are used to check the configuration and status of resources:

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

status_*
Provides summary information about the state of resources.

Managing
The following types of commands are used to manage resources:

start_*, init_*, or activate_*
Explicitly starts a configured resource. Some resources can be activated implicitly as a result of activating other resources.

stop_*, term_*, or deactivate_*
Explicitly stops a resource. Some resources can be stopped implicitly (for example, as a result of a period of inactivity).

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

Administrative commands are listed alphabetically in Chapter 2, "Administration Commands," on page 11.

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

- On Windows® clients there is no **snaadmin** program, so no commands can be issued.

- On AIX® and Linux clients you can issue any **query** or **status** command. Some other administration commands, defined in Chapter 2, "Administration Commands," on page 11, explicitly say that they can be issued from an IBM Remote API Client. Otherwise these commands are available only from a server.

Using snaadmin

Before you can use the snaadmin program, Communications Server for Linux must be started. If Communications Server for Linux has not been started, enter the following command on the Linux command line:

sna start

You can use **snaadmin** to configure and manage Communications Server for Linux. Use **snaadmin** as an alternative to the Motif administration program when any of the following is true:

- You want to configure resource parameters that are not frequently used.
- You do not have X display capabilities.

When you issue a command or use the Motif administration program, Communications Server for Linux changes the configuration file. For more information about configuration files, see Appendix B, "Configuration Files," on page 593.

For more information about using the Motif administration program, refer to *Communications Server for Linux Administration Guide*.

Use the following syntax for **snaadmin**:

snaadmin [**-n**
node] [**-d**] [**-a**] [
-h]
<**-i** *infile*> |
<*command*,
parameter1=*value1*
, *parameter2*=*value2*, ...>

For information about the options you can use on the command line, see "Command Line Options." For information about parameter syntax, see "Parameter Syntax Used for Administration Commands" on page 4.

Administrative commands are listed alphabetically in Chapter 2, "Administration Commands," on page 11, and include descriptions containing the following:

- Purpose of the command
- Whether the command can be issued to an active node or an inactive node or to the domain configuration file
- Other commands that must precede it
- Details about the parameters for the command, including parameter types and default values
- Returned information

Command Line Options

You can use one or more of the following options when you use the **snaadmin** program:

-n *node*

Sends the command to the named node. By default, node commands are sent to the local node.

The node name is a string of 1–128 characters. If it includes a **.** (period) character, Communications Server for Linux assumes that it is a fully-qualified name; otherwise it performs a DNS lookup to determine the node name.

-d List detailed information.

-a List all items (**query_*** commands only).

You do not need to specify **-a** to list all resources of a particular type. The **-a** option is implied by default if no particular resource is named.

-h Display help. For more information, see "Using Command Line Help" on page 4.

-c Change a particular parameter (available on a select number of **define_** commands only). For more information, see "Changing Specific Parameters."

-i *infile*

Use commands from the named file. This must be a file in the Communications Server for Linux configuration file format (as described in Appendix B, "Configuration Files," on page 593), and not just a list of commands and parameters as you would enter them on the command line.

Changing Specific Parameters

The command line option **-c** enables the user to change a specific parameter on an existing command without reentering the entire command. Specify the command name and parameter to be changed on the command line. This option is supported only for the following commands (an error message is returned for all other commands):

- **define_cpic_side_info**
- **define_downstream_lu**
- **define_ethernet_dlc**
- **define_ethernet_ls**
- **define_ethernet_port**
- **define_ip_dlc**
- **define_ip_ls**
- **define_ip_port**
- **define_local_lu**
- **define_lu_0_to_3**
- **define_mode**
- **define_mpc_dlc**
- **define_mpc_ls**
- **define_mpc_port**
- **define_node**
- **define_partner_lu**
- **define_qllc_dlc**
- **define_qllc_ls**
- **define_qllc_port**
- **define_sdlc_dlc**

- **define_sdlc_ls**
- **define_sdlc_port**
- **define_tp**
- **define_tr_dlc**
- **define_tr_ls**
- **define_tr_port**
- **define_userid_password**

Using Command Line Help

Help is available using the **-h** and **-d** options on **snaadmin** as follows:

snaadmin -h	Displays general information about administration commands and instructions for specifying commands and parameters on the command line.
snaadmin -h -d	Lists all administration commands.
snaadmin -h *command*	Displays a description of the specified command.
snaadmin -h -d *command*	Displays a description of the specified command and lists the parameters for this command.

Parameter Syntax Used for Administration Commands

Use the syntax described in the following sections to specify parameters in administration commands. The information included in these sections applies to both configuration files and **snaadmin** commands, except where indicated.

The parameters in a command can be specified in any order, except as noted in "Subrecords in Administration Commands" on page 6.

Parameter Types

Each parameter in an administration command is one of the following types:

Character

A character string entered using locally displayable characters (for example the *lu_name* parameter value). The individual parameter descriptions give details of the maximum and minimum length of each string. The parameter descriptions also indicate when the string must consist of characters from a particular character set (such as alphanumeric, type-A, or Linux file name characters). Otherwise, you can use any character that is displayable in your computer's local character set. Character strings are case-sensitive.

If you enter a character string as command-line input to the **snaadmin** program and the string contains one or more commas, you must precede each comma with % so that the program does not interpret the comma as a separator between parameters. To enter a % character by itself, use two % characters—%%. (This appears as %% in configuration files and in text output from administration commands, but is interpreted as a single character.)

A name parameter entered as a character string that starts with the @ character is reserved and should not be used. An exception is the *resource_name* parameter on the **add_dlc_trace** and **remove_dlc_trace** commands. Also, avoid using names that start with the $ character because you may overwrite a name reserved for the system. Currently, all names starting with $G are used by the system.

Do not enclose character strings in quotation marks (""). If you need to include quotation marks within a character string, the following restrictions apply:

- The " character must be a valid character for the parameter you are defining.
- The string must contain an even number of quotation marks.
- Each quotation mark must be preceded by a backslash character, \", to avoid interpretation by the Linux shell.

Decimal

A numeric value (for example 128). The individual parameter descriptions give details of the maximum and minimum values. Specify numeric values in decimal, not in hexadecimal, unless the values are explicitly defined as hex numbers.

Hex number

A numeric value given in hexadecimal, specified as an even number of hexadecimal digits preceded by 0x (for example 0xF0). A hex number is normally one byte, specified as two hexadecimal digits, unless a length is explicitly specified; for example, *sense_data* on **deactivate_session** is defined to have a length of 4 (a four-byte value specified as eight hexadecimal digits).

The individual parameter descriptions give details of the maximum and minimum values or any other restrictions on the valid values if applicable. If no restrictions are noted, you can use any value. The characters A–F are not case-sensitive; you can use either uppercase or lowercase.

Hex array

An array of hexadecimal digits, which can be represented either by enclosing the digits in angle brackets (for example <010A0B0C>) or by preceding the digits with 0x. The individual parameter descriptions give details of the maximum and minimum length of the array, and any restrictions on its value. The characters A–F are not case-sensitive; you can use either uppercase or lowercase.

If you are entering a hex array as command-line input to the **snaadmin** program, you must precede each angle bracket by a backslash character (\< or \>), to avoid interpretation by the Linux shell.

Constant

One of two or more defined values, specified as an ASCII character string without quotation marks (for example PRIMARY). Defined constants are used for parameters that have a fixed set of valid values, such as PRIMARY / SECONDARY / NEGOTIABLE; the individual parameter descriptions list the defined values for each parameter. Defined constants are not case-sensitive; you can use either uppercase or lowercase.

The command descriptions list the type for each parameter.

Default Parameter Values

Some administration command parameters, such as the name of the resource you are defining or starting, must always be explicitly specified. For other parameters, Communications Server for Linux provides default values. For a standard configuration, you do not need to specify every parameter in a command. The individual parameter descriptions include information about default values where applicable. If no default value is shown for a parameter, you must specify it explicitly.

The default parameter values used for administration command parameters can be different from the default values used for the Motif administration program.

Blank Space

Embedded space characters are valid within character strings only when the string's character set allows them, and are not valid within any other type of parameter value. For example:

- The character string `LU001` is valid for the *lu_name* parameter.
- The character string `LU 001` is valid for the *description* parameter which allows any characters (including spaces), but not for the *lu_name* parameter which does not allow space characters.
- The hex array `<01020304>` is valid.
- The hex array `<01 02 03 04>` is not valid.

All blank space before or after descriptors, parameter names, or parameter values (that is, any combination of spaces and tabs) is ignored.

There is no need to use quotation marks (`""`) around parameter values containing spaces.

Subrecords in Administration Commands

Some administration commands include data whose format can vary between instances of the command. To allow for this, the variable data is specified in optional subrecords. This means that a command consists of a series of parameters common to all instances of that command type, followed by subrecords containing the variable data.

All commands have the following order:

1. The *command_name*
2. Common parameters

All commands that have one or more subrecords have the following order:

1. The *command_name*
2. Common parameters
3. A `subrecord_name`, in braces { }
4. Parameters associated with that `subrecord_name`
5. Further instances of the `subrecord_name`, each followed by the parameters associated with it

In a configuration file, each of these names and parameters is on a separate line; in a command issued to **snaadmin**, they are separated by commas.

All the parameters associated with the *command_name* (and not with a subrecord) must be listed after the *command_name* and before the first `subrecord_name`; all the parameters associated with a particular `subrecord_name` must be listed after that `subrecord_name` and before the next `subrecord_name`, if any, or the next *command_name*. However, the order of individual parameters within a subrecord (or within the common parameters) is not important.

List Options on query_* Commands

You can obtain information about Communications Server for Linux resources by issuing a **query**_* command for the appropriate resource type. For example, you can obtain information about the configuration and status of an LS by issuing

query_ls. A **query**_* command can return information about a specific resource (for example, the configuration of a particular LS) or about multiple resources of the same type (for example, information about all configured link stations), depending on the options used. In addition, some **query**_* commands have the option of returning either summary or detailed information about the specified resources.

Note: Most users will not need to use the *num_entries* and *list_options* parameters described in this section. Instead, you can use the command-line options **-a** and **-d** with the **snaadmin** command to specify which entry or entries you want and the level of detail you need:

- To return a single named entry, specify the resource name of the entry you want, and do not specify the **-a** option.
- To return all entries, specify the **-a** option and do not specify a resource name.
- To return detailed information (either for a single named entry or for multiple entries), add the **-d** option to the command.

For more information about these options, see "Command Line Options" on page 2.

Obtaining Information About a Single Resource or Multiple Resources: You can think of the information returned by **query**_* commands as being stored in the form of a list, ordered according to the name of the resource. For example, the information returned by **query_ls** is in order of LS name. The normal order of the list is as follows:

- By name length (shortest name first)
- By ASCII lexicographical ordering for names of the same length

Individual command descriptions note when the list ordering differs from the preceding order (for example, when the list is ordered by a numeric value).

You can obtain information about multiple resources by requesting the complete list or a specified part of it. The following parameters on the **query**_* command determine which entries from the list are returned:

num_entries
: Maximum number of resources for which information should be returned. You can specify `1` to return a specific entry, a number greater than `1` to return multiple entries, or `0` (zero) to return all entries. The default is to return all entries if you specify only the name of the query command and do not specify *num_entries* or the resource name, or to return one entry if you do not specify *num_entries* but do specify the resource name.

list_options
: The position in the list of the first entry required, specified by one of the following options:

  `FIRST_IN_LIST`
  : First entry in the list

  `LIST_INCLUSIVE`
  : Entries starting from a specific named entry

  `LIST_FROM_NEXT`
  : Entries starting from the next entry after a specific named entry. The name specified gives the starting position according to the list ordering; the name need not exist in the list. For example, if the list

contains entries NODEA, NODEB, NODED, NODEF, and the application requests entries starting from the first entry after NODEC, the first entry returned is NODED.

If the *list_options* parameter is set to `LIST_INCLUSIVE` or `LIST_FROM_NEXT`, another parameter on the command specifies the name of an entry in the list that gives the starting position for the required entries. The description of *list_options* in each command description explains which parameter is used to identify the starting position. If you specify one of these options but do not specify the parameter giving the starting position, the *list_options* parameter is ignored and the returned information starts from the first entry in the list.

To request all entries in the list when using the **snaadmin** program, you can use the command-line option **-a** instead of specifying *num_entries* as `0` and *list_options* as `FIRST_IN_LIST` (also, the default is to return all entries when *num_entries* and the resource name are not specified). This option returns all entries unless you explicitly set *num_entries* or *list_options* to return specific entries.

The number of entries returned is the smaller of the following values:

- The *num_entries* parameter, if this is nonzero
- The number of entries between the specified starting position and the end of the list

Obtaining Summary or Detailed Information: Some **query**_* commands provide the option of returning summary or detailed information about the specified resources. For example, **query_local_lu** can return only the LU name, LU alias and description (summary information), or it can also return additional information such as the LU address and session limit (detailed information). The description of each **query**_* command indicates whether the command includes the option of returning summary or detailed information.

For the commands that provide the summary or detailed option, use the *list_options* parameter to indicate whether summary or detailed information is required, as well as the starting position within the list. To specify these options, combine two values with a + character (one value to specify whether summary or detailed information is required, and one value to specify the starting position in the list), and set the *list_options* parameter to the combination of these two values. For example, to specify summary information for all DLCs defined at the node, supply the value `SUMMARY+FIRST_IN_LIST` for the *list_options* parameter on the **query_dlc** command.

To request detailed information, you can use the **-d** option on the **snaadmin** command line instead of specifying a value of `DETAIL` for the *list_options* parameter. The **-d** option returns detailed information unless you explicitly specify a value of `SUMMARY` for the *list_options* parameter which returns summary information only.

Examples of Administration Commands

This section provides some examples of **snaadmin** commands as you would issue them on the command line. Many of the parameters in these commands are not specified, so **snaadmin** uses the default values; see the description of each command in Chapter 2, "Administration Commands," on page 11 for details of the default parameter values.

The following commands define connectivity to a remote system over Ethernet. Note the use of angle brackets to specify the *mac_address* parameter as a

hexadecimal array. Each angle bracket needs to be preceded by a backslash character \ to prevent interpretation by the Linux shell.

```
snaadmin define_ethernet_dlc, dlc_name = DLCNAME, initially_active = YES
snaadmin define_ethernet_port, port_name = PORTNAME, dlc_name = DLCNAME, initially_active = YES
snaadmin define_ethernet_ls, ls_name = LSNAME1, port_name = PORTNAME, mac_address = \<000000000000\>
```

The following command defines access for a TN3270 client. Note the use of brace characters to specify the TN3270 session data. Each brace character needs to be preceded by a backslash character \ to prevent interpretation by the Linux shell.

```
snaadmin define_tn3270_access, default_record=YES, description="Test client", \{tn3270_session_data\}, port_number=8001
```

The following commands define a local LU used for LU6.2, and the partner LU that it communicates with.

```
snaadmin define_local_lu, lu_name=LUNAME1, lu_alias=LUNAME1
snaadmin define_partner_lu, fqplu_name=APPN.PTNRLU, plu_alias=PTNR01
```

The following command activates a session between the local LU and partner LU, using the standard SNA mode named #CONNECT. Note the use of a backslash character \ before the # character in this name, to prevent interpretation by the Linux shell.

```
snaadmin activate_session, lu_alias=LUNAME1, plu_alias=PTNR01, mode_name=\#INTER
```

The following commands request information about the definition of the partner LU, its current status, and the sessions between the local and partner LUs. In all cases, the -d or DETAIL value is used to request detailed information, overriding the default which is to provide summary information only.

```
snaadmin -d query_partner_lu_definition, plu_alias=PTNR01
snaadmin -d query_partner_lu, lu_name=LUNAME1, plu_alias=PTNR01
snaadmin query_session, num_entries=0, list_options=DETAIL+FIRST_IN_LIST, lu_name=LUNAME1, plu_alias=PTNR01
```

Chapter 2. Administration Commands

This chapter provides reference information about administration commands used for configuring, defining, deleting, querying, checking status, starting, and stopping the following resources: local nodes, connectivity components, directory entries, network topology (query only), type 0–3 LUs and pools. The commands are listed in alphabetical order.

activate_session

The **activate_session** command requests Communications Server for Linux to activate a session between the local LU and a specified partner LU, using a specified mode. You must issue an **initialize_session_limit** command before issuing an **activate_session** command unless the *cnos_permitted* parameter is set to YES.

This command must be issued to a running node.

This command can be issued from a client. If it is issued from an AIX or Linux client, the command must run with the userid root, or with a userid that is a member of the sys group (AIX) or sna group (Linux).

Supplied Parameters

Parameter name	Type	Length	Default
[activate_session]			
lu_name	character	8	(null string)
lu_alias	character	8	(null string)
plu_alias	character	8	(null string)
mode_name	character	8	
fqplu_name	character	17	(null string)
polarity	constant		POL_EITHER
cnos_permitted	constant		YES

Supplied parameters are:

lu_name
: LU name of the local LU. This name is a type-A character string. To indicate that the LU is defined by its LU alias instead of its LU name, do not specify this parameter.

lu_alias
: LU alias of the local LU. This alias is a character string using any locally displayable characters. This parameter is used only if *lu_name* is not specified.

  If *lu_name* and *lu_alias* are not specified, the command is forwarded to the LU associated with the CP (the default LU).

plu_alias
: LU alias of the partner LU. This alias is a character string using any locally displayable characters. To indicate that the partner LU is defined by its fully qualified LU name instead of its LU alias, do not specify this parameter.

mode_name
Name of the mode to be used by the LUs. This name is a type-A character string starting with a letter.

fqplu_name
Fully qualified name of the partner LU. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character partner LU name.

This parameter is used only if the *plu_alias* parameter is not specified; it is ignored if *plu_alias* is specified.

polarity
The polarity for the session. Possible values are:

```
POL_EITHER
POL_FIRST_SPEAKER
POL_BIDDER
```

If `POL_EITHER` is set, **activate_session** activates a first speaker session if available, otherwise a bidder session is activated. If `POL_FIRST_SPEAKER` or `POL_BIDDER` is set, **activate_session** only succeeds if a session of the requested polarity is available.

cnos_permitted
Indicates that CNOS processing is permitted. Possible values are:

YES CNOS processing is permitted.

NO CNOS processing is not permitted.

If the activation of a new session is not possible because the session limits for the specified mode are reset, and this parameter is set to `YES`, implicit CNOS processing will initialize the session limits. Execution of this command is suspended while CNOS processing is active.

Returned Parameters

If the command executes successfully, the following parameters are returned:

primary_rc
`OK`

secondary_rc
Possible values are:

AS_NEGOTIATED
The session was activated successfully; the session limit defined for the mode was negotiated during the activation process.

AS_SPECIFIED
The session was activated successfully; the session limit was not changed.

session_id
The session ID of the new session.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc

PARAMETER_CHECK

secondary_rc

Possible values are:

EXCEEDS_MAX_ALLOWED

The session cannot be activated because this would exceed the current session limit for this LU-LU-mode combination.

INVALID_LU_ALIAS

The *lu_alias* parameter did not match any defined local LU alias.

INVALID_LU_NAME

The *lu_name* parameter did not match any defined local LU name.

INVALID_PLU_NAME

The *fqplu_name* parameter did not match any defined partner LU name, or the *plu_alias* parameter did not match any defined partner LU name.

INVALID_CNOS_PERMITTED

The value specified in the *cnos_permitted* parameter was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

If the command does not execute because other conditions exist, Communications Server for Linux returns the following parameters:

primary_rc

ACTIVATION_FAIL_NO_RETRY

The session could not be activated because of a condition that requires action (such as a configuration mismatch or a session protocol error). Do not retry activating the session. Check the Communications Server for Linux log file for information about the error condition and correct it before retrying.

ACTIVATION_FAIL_RETRY

The session could not be activated because of a temporary condition (such as a link failure). Retry, preferably after a timeout to allow the condition to clear. Check the Communications Server for Linux log file for information about the error condition.

secondary_rc

A secondary return code is not returned.

Appendix A, "Common Return Codes from snaadmin Commands," on page 589 lists combinations of primary and secondary return codes that are common to all commands.

add_backup

The **add_backup** command adds a server to the list of backup master servers in the **sna.net** file so this server can act as the master configuration file server if the current master becomes inactive. The new server is added to the end of the list; it will only become the master server if all other servers listed in the file are inactive.

This command must be issued without specifying a node name.

Supplied Parameters

Parameter name	Type	Length
[add_backup]		
backup_name	character	128

Supplied parameter is:

backup_name
The name of the server to be added to the list of backup servers.

If the server name includes a **.** (period) character, Communications Server for Linux assumes that it is a fully-qualified name; otherwise it performs a DNS lookup to determine the server name.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc

DUPLICATE_RECORD
The server name specified in the *backup_name* parameter is already listed in the file.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

add_dlc_trace

The **add_dlc_trace** command controls tracing on SNA messages sent on a DLC. It can be used to activate tracing on a particular DLC, port, LS, or HPR RTP connection. It can also be used to activate tracing on a particular session on a specified LS or to specify which types of messages are to be traced. The command can also be used to activate tracing on all DLCs, ports, link stations, and HPR RTP connections. For more information about how to use Communications Server for Linux tracing, refer to the *Communications Server for Linux Diagnostics Guide*.

If multiple **add_dlc_trace** commands relating to the same resource are issued, a message will be traced if it matches any of the commands currently active. For example:

- If you issue a command to trace all messages for a port and its link stations and then issue a second command to trace only messages with a specified LFSID for one of the link stations owned by the port, all messages for the LS will continue to be traced (because they match the first command). If you then use **remove_dlc_trace** to remove tracing for the port, messages on the LS with the specified LFSID will continue to be traced (because they match the second command which is still active), but other messages on this LS will not be traced.
- If you issue a command to trace XID messages on all resources and then issue a second command to trace SC and DFC messages on a particular LS, all three message types will be traced for this LS.

If you are tracing an SDLC line and would like more detailed trace information, you can get this by using internal tracing on SDLC as well as line tracing. The additional detail is formatted as part of the output for line tracing, so that you will see all of the SDLC tracing in one file. For more information, see "set_trace_type" on page 561.

Note: The **set_trace_type** command includes an option to truncate each entry in trace files to a specified length. This option applies to DLC tracing as well as to the kernel component tracing specified by **set_trace_type**.

Supplied Parameters

Parameter name	Type	Length	Default
[add_dlc_trace]			
resource_type	constant		ALL_RESOURCES
resource_name	character	8	(null string)
sidh	hex byte		0
sidl	hex byte		0
odai	constant		NO
message_type	constant		TRACE_ALL

Supplied parameters are:

resource_type
Specifies the resource to be traced and optionally the specific message types to be traced for this resource. Possible values are:

ALL_RESOURCES
Specify tracing options for all DLCs, ports, link stations, and RTP connections.

DLC Specify tracing options for the DLC named in *resource_name* and for all ports and link stations that use this DLC.

PORT Specify tracing options for the port named in *resource_name* and for all link stations that use this port.

LS Specify tracing options for the LS named in *resource_name*.

RTP Specify tracing options for the RTP connection named in *resource_name*.

PORT_DEFINED_LS
Specify tracing options for the port named in *resource_name* and for all defined link stations (but not implicit link stations) that use this port.

PORT_IMPLICIT_LS
Specify tracing options for the port named in *resource_name* and for all implicit link stations (but not defined link stations) that use this port.

resource_name
The name of the DLC, port, LS, or RTP connection for which tracing is to be activated. Do not specify this parameter if *resource_type* is set to `ALL_RESOURCES`.

If *resource_type* is set to `RTP`, you can specify the name of a particular RTP connection (this name begins with the `@` character), or you can omit this parameter to indicate that all RTP traffic is to be traced.

The following three parameters identify the Local Form Session Identifier (LFSID) for a session on the specified LS. These parameters are valid only if *resource_type* is set to `LS`, and they indicate that only messages on this session are to be traced. The LFSID consists of the following parameters:

sidh Session ID high byte

sidl Session ID low byte

odai Origin Destination Assignor Indicator. Possible values are:

YES The BIND sender is the node containing the secondary link station.

NO The BIND sender is the node containing the primary link station.

message_type
The type of messages to trace for the specified resource or session. To trace all messages, set this parameter to `TRACE_ALL`. To trace specific messages, specify one or more of the following values (combined using a `+` character):

TRACE_XID
Trace XID messages

TRACE_SC
Trace Session Control Request/Response Units (RUs)

TRACE_DFC
Trace Data Flow Control RUs

TRACE_FMD
Trace Function Management Data messages

TRACE_SEGS
Trace Non-BBIU segments that do not contain an RH

TRACE_CTL
Trace Messages other than MUs and XIDs

TRACE_NLP
Trace Network-Layer Protocol messages

TRACE_NC
Trace Network Control messages

For tracing on an RTP connection, the values TRACE_XID, TRACE_NLP, and TRACE_CTL are ignored. At least one of the other values listed must be specified for RTP tracing.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_RESOURCE_TYPE
The value specified in the *resource_type* parameter was not valid.

INVALID_MESSAGE_TYPE
The value specified in the *message_type* parameter was not valid.

INVALID_RTP_CONNECTION
The *resource_name* parameter does not match any RTP connection.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

aping

The **aping** command is the APPN version of the "ping" utility and allows a management application to check the communications path from a local LU to a remote LU in the network.

Instead of using the **aping** command, you can use the **APING** program described in *Communications Server for Linux APPC Application Suite User's Guide*.

Communications Server for Linux **aping** is implemented using an internally-defined APPC TP. This TP sends data to the partner LU, and optionally

receives data from the partner LU. If the TP completes successfully, **aping** returns information about the time taken to allocate a conversation to the partner LU and to send and receive data.

This command is intended for checking the path to an LU on a remote node. Using **aping** to check communications with a partner LU on the local node will impact the performance of other programs on the local computer, and is not recommended.

This command must be issued to a running node.

Supplied Parameters

Parameter name	Type	Length	Default
[aping]			
lu_name	character	8	(null string)
lu_alias	character	8	(null string)
plu_alias	character	8	(null string)
mode_name	character	8	
tp_name	character	64	APINGD
security	constant		NONE
pwd	character	10	(null string)
user_id	character	10	(null string)
dlen	decimal		0
consec	decimal		1
fqplu_name	character	17	(null string)
echo	constant		NO
iterations	decimal		0
partner_ver_len	decimal		0

Supplied parameters are:

lu_name
: LU name of the local LU. This name is a type-A character string. To indicate that the LU is identified by its LU alias instead of its LU name, do not specify this parameter.

lu_alias
: LU alias of the local LU. This parameter is used only if *lu_name* is not specified. If *lu_name* and *lu_alias* are not specified, the command is forwarded to the LU associated with the CP (the default LU).

plu_alias
: Partner LU alias. This parameter should be the alias of an LU on a remote node; using *aping* with a partner LU on the local node is not recommended.

  To indicate that the LU is identified by its fully qualified name instead of its alias, do not specify this parameter, but specify the LU name in the *fqplu_name* parameter.

mode_name
: Name of the mode used by the LU pair. This name is a type-A character string starting with a letter.

tp_name
: Name of the invoked TP. This parameter is generally set to "APINGD."

security
: Specifies whether conversation security information is required to start the TP. Possible values are:

  NONE No security information is required.

SAME Security information can be verified by the TP that invoked this TP on behalf of a third TP.

PGM A password and user ID are required to start the TP. The password is sent unencrypted if password substitution is not supported on the session. If password substitution is supported on the session, the password is sent encrypted.

PGM_STRONG
A password and user ID are required to start the TP, but the password must not be sent in clear text. If password substitution is not supported on the session, the **aping** fails. Otherwise, the password is sent encrypted.

pwd Password required to access the partner TP. This parameter is required only if the *security* parameter is set to PGM or PGM_STRONG. This password is a type-AE character string.

user_id User ID required to access the partner TP. This parameter is required only if the *security* parameter is set to SAME, PGM or PGM_STRONG. This ID is a type-AE character string.

dlen Length of the data string to be sent to the partner LU. (You do not need to provide a data string; the APING TP simply sends a string of zeros of the specified length.) Specify a value in the range 0–65,535.

consec Number of consecutive data strings sent to the partner LU before a response is required. The APING TP sends this number of data strings, with each string containing the number of bytes specified by the *dlen* parameter. The APING TP then requests either data or a confirmation message from the partner TP, depending on the setting of the *echo* parameter. Specify a value in the range 1–65,535.

fqplu_name
Fully qualified name of the partner LU. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character partner LU name.

This name should be the name of an LU on a remote node; using **aping** with a partner LU on the local node is not recommended.

This parameter is used only if the *plu_alias* parameter is not specified; it is ignored if *plu_alias* is specified.

echo Specifies whether the APING TP receives data or requests confirmation from the partner LU after sending data to it. Possible values are:

YES After sending the specified number of data strings, the APING TP receives data from the partner LU.

NO After sending the specified number of data strings, the APING TP requests confirmation from the partner LU, but does not receive data.

iterations
Number of times that the APING TP performs the sequence of sending data to the partner LU and requesting either data or confirmation. Specify a value in the range 0–65,535.

partner_ver_len
Maximum length of the partner TP verification data string to return. Specify a value in the range 0–3000.

Returned Parameters

Parameter name	Type	Length
alloc_time	decimal	
min_time	decimal	
avg_time	decimal	
max_time	decimal	
partner_ver_len	decimal	
partner_ver_data	hex array	(max_length as specified on command)

If the command executes successfully, Communications Server for Linux returns the following parameters:

alloc_time
The time in milliseconds to allocate a conversation to the partner—the time taken for the MC_ALLOCATE verb issued by the APING TP to complete.

min_time
The minimum time in milliseconds required for a data-sending iteration—the shortest measured time for a single iteration of sending data and receiving either data or confirmation. If *iterations* was set to `0` (zero), this parameter is not used.

avg_time
The average time in milliseconds required for a data-sending iteration—the average time for a single iteration of sending data and receiving either data or confirmation. If *iterations* was set to `0` (zero), this parameter is not used.

max_time
The maximum time in milliseconds required for a data-sending iteration—the longest measured time for a single iteration of sending data and receiving either data or confirmation. If *iterations* was set to `0` (zero), this parameter is not used.

partner_ver_len
Actual length of the verification string returned by the partner TP.

partner_ver_data
Verification string returned by the partner TP. If *partner_ver_len* is `0` (zero), then this string is not returned.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
`PARAMETER_CHECK`

secondary_rc
Possible values are:

`INVALID_LU_ALIAS`
The *lu_alias* parameter value did not match any defined LU alias.

`INVALID_LU_NAME`
The *lu_name* parameter value did not match any defined LU name.

BAD_PARTNER_LU_ALIAS
The value specified for *plu_alias* did not match any defined partner LU.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

If the command does not execute because other conditions exist, Communications Server for Linux returns the following parameters:

primary_rc

ALLOCATION_ERROR
Communications Server for Linux could not allocate the APPC conversation with the remote TP.

secondary_rc
Possible values are:

ALLOCATION_FAILURE_NO_RETRY
The conversation could not be allocated because of a permanent condition, such as a configuration error or session protocol error. Check the *sense_data* parameter and the error log file for more information. Do not attempt to retry the **aping** command until the error has been corrected.

ALLOCATION_FAILURE_RETRY
The conversation could not be allocated because of a temporary condition, such as a link failure. Check the error log file for more information. Retry the **aping** command, preferably after a timeout to allow the condition to clear.

SECURITY_NOT_VALID
The user ID or password specified was not accepted by the partner LU.

TP_NAME_NOT_RECOGNIZED
The partner LU does not recognize the specified TP name.

TRANS_PGM_NOT_AVAIL_NO_RETRY
The remote LU rejected the allocation request because it was unable to start the requested partner TP. The condition is permanent. The reason for the error may be logged on the remote node. Do not retry the **aping** command until the cause of the error has been corrected.

TRANS_PGM_NOT_AVAIL_RETRY
The remote LU rejected the allocation request because it was unable to start the requested partner TP. The condition may be temporary, such as a timeout. The reason for the error may be logged on the remote node. Retry the **aping** command, preferably after a timeout to allow the condition to clear.

sense_data
If the *secondary_rc* parameter is ALLOCATION_FAILURE_NO_RETRY, this parameter contains the SNA sense data associated with the error. For all other *secondary_rc* values, this parameter is not returned.

primary_rc

CONV_FAILURE_NO_RETRY
The APPC conversation with the partner TP was terminated because of a permanent condition, such as a session protocol error. Check the error log file to determine the cause of the error. Do not retry the **aping** command until the error has been corrected.

primary_rc

CONV_FAILURE_RETRY
The APPC conversation with the partner TP was terminated because of a temporary error. Retry the **aping** command. If the problem occurs again, check the error log file to determine the cause of the error.

primary_rc

DEALLOC_ABEND
The partner TP deallocated the APPC conversation because of an error condition. The reason for the error may be logged on the remote node.

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

change_session_limit

The **change_session_limit** command requests Communications Server for Linux to change the session limits for a particular LU-LU-mode combination. Sessions can be activated or deactivated as a result of processing this command.

This command must be issued to a running node.

Supplied Parameters

Parameter name [change_session_limit]	Type	Length	Default
lu_name	character	8	(null string)
lu_alias	character	8	(null string)
plu_alias	character	8	(null string)
fqplu_name	character	17	(null string)
mode_name	character	8	
set_negotiable	constant		NO
plu_mode_session_limit	decimal		
min_conwinners_source	decimal		0
min_conwinners_target	decimal		0
auto_act	decimal		0
responsible	constant		SOURCE

Supplied parameters are:

lu_name
LU name of the local LU. This name is a type-A character string. To indicate that the LU is defined by its LU alias instead of its LU name, do not specify this parameter.

lu_alias
LU alias of the local LU. This alias is a character string using any locally displayable characters. It is used only if *lu_name* is not specified.

If *lu_name* and *lu_alias* are not specified, the command is forwarded to the LU associated with the CP (the default LU).

plu_alias
LU alias of the partner LU. This alias is a character string using any locally displayable characters. To indicate that the partner LU is defined by its fully qualified LU name instead of its LU alias, do not specify this parameter.

fqplu_name
Fully qualified name of the partner LU. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character partner LU name.

This parameter is used only if the *plu_alias* parameter is not specified; it is ignored if *plu_alias* is specified.

mode_name
Name of the mode to be used by the LUs. This name is a type-A character string starting with a letter.

set_negotiable
Specifies whether the maximum negotiable session limit for this mode, as defined by **define_mode**, should be modified. Possible values are:

YES Use the value specified by *plu_mode_session_limit* as the maximum negotiable session limit for this LU-LU-mode combination.

NO Leave the maximum negotiable session limit as the value specified for the mode.

plu_mode_session_limit
Requested total session limit for this LU-LU-mode combination—the maximum number of parallel sessions allowed between these two LUs using this mode. This value can be negotiated with the partner LU. Specify a value in the range 1–32,767 (which must not exceed the session limit specified for the local LU on the **define_local_lu** command). To specify a value of 0 (zero) use the **reset_session_limit** command.

min_conwinners_source
Minimum number of sessions using this mode for which the local LU is the contention winner. The sum of the *min_conwinners_source* and *min_conwinners_target* parameters must not exceed the *plu_mode_session_limit* parameter. Specify a value in the range 0–32,767.

min_conwinners_target
Minimum number of sessions using this mode for which the partner LU is the contention winner. The sum of the *min_conwinners_source* and *min_conwinners_target* parameters must not exceed the *plu_mode_session_limit* parameter. Specify a value in the range 0–32,767.

auto_act
Number of sessions to activate automatically after the session limit is changed. The actual number of automatically activated sessions is the minimum of this value and the negotiated minimum number of contention winner sessions for the local LU. When sessions are deactivated normally (specifying DEACT_NORMAL for the *type* parameter on **deactivate_session**) this limit, new sessions are activated up to this limit. Specify a value in the range 0–32,767 (which must not exceed the *plu_mode_session_limit* parameter or the session limit specified for the local LU on the **define_local_lu** command).

responsible
Indicates whether the local LU or partner LU is responsible for deactivating sessions after the session limit is changed. Possible values are:

SOURCE The local LU is responsible for deactivating sessions after the session limit has been changed.

TARGET The partner LU is responsible for deactivating sessions after the session limit has been changed.

Returned Parameters

If the command executes successfully, the following parameters are returned:

primary_rc
OK

secondary_rc
Possible values are:

AS_NEGOTIATED
The session limits were changed, but one or more values were negotiated by the partner LU.

AS_SPECIFIED
The session limits were changed as requested, without being negotiated by the partner LU.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

EXCEEDS_MAX_ALLOWED
The *plu_mode_session_limit*, *min_conwinners_source*, *min_conwinners_target*, or *auto_act* parameter was set to a value outside the valid range.

CANT_CHANGE_TO_ZERO
The *plu_mode_session_limit* parameter cannot be set to 0 (zero) using this command; use the **reset_session_limit** command instead.

INVALID_LU_ALIAS
The *lu_alias* parameter did not match any defined local LU alias.

INVALID_LU_NAME
The *lu_name* parameter did not match any defined local LU name.

INVALID_MODE_NAME
The *mode_name* parameter did not match any defined mode name.

INVALID_PLU_NAME
The *fqplu_name* parameter did not match any defined partner LU name.

INVALID_RESPONSIBLE
The *responsible* parameter was not set to a valid value.

INVALID_SET_NEGOTIABLE
The *set_negotiable* parameter was not set to a valid value.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc

MODE_RESET
No sessions are currently active for this LU-LU-mode combination. Use **initialize_session_limit** instead of **change_session_limit** to specify the limits.

Other Conditions

If the command does not execute because other conditions exist, Communications Server for Linux returns the following parameters:

primary_rc
ALLOCATION_ERROR

secondary_rc

ALLOCATION_FAILURE_NO_RETRY
A session could not be allocated because of a condition that requires action. Check the *sense_data* parameter and logged messages to determine the reason for the failure, and take any action required. Do not attempt to retry the command until the condition has been corrected.

sense_data
If the *secondary_rc* parameter is ALLOCATION_FAILURE_NO_RETRY, this parameter contains the SNA sense data associated with the error. For all other *secondary_rc* values, this parameter is not returned.

primary_rc

CONV_FAILURE_NO_RETRY
The session limits could not be initialized because of a condition that requires action (such as a configuration mismatch or a session protocol error). Check the Communications Server for Linux log file for information about the error condition, and correct it before retrying this command.

primary_rc
CNOS_PARTNER_LU_REJECT

secondary_rc

CNOS_COMMAND_RACE_REJECT
The command failed because the specified mode was being accessed by another administration program (or internally by the

Communications Server for Linux software) for session activation or deactivation, or for session limit processing. Retry the command.

Appendix A, "Common Return Codes from snaadmin Commands," on page 589 lists combinations of primary and secondary return codes that are common to all commands.

deactivate_conv_group

The **deactivate_conv_group** command requests Communications Server for Linux to deactivate the session corresponding to the specified APPC conversation group. Although this command is available within the command-line administration program, deactivating a session identified by its conversation group is more usually done using the NOF verb DEACTIVATE_CONV_GROUP from within an APPC TP. The conversation group identifier is returned by the APPC verbs [MC_]ALLOCATE, [MC_]GET_ATTRIBUTES, and RECEIVE_ALLOCATE.

This command must be issued to a running node.

Supplied Parameters

Parameter name	Type	Length	Default
[deactivate_conv_group]			
lu_name	character	8	(null string)
lu_alias	character	8	(null string)
conv_group_id	decimal		
type	constant		DEACT_NORMAL
sense_data	hex number	4	0x0

Supplied parameters are:

lu_name
LU name of the local LU. This name is a type-A character string. To indicate that the LU is defined by its LU alias instead of its LU name, do not specify this parameter.

lu_alias
LU alias of the local LU. This alias is a character string using any locally displayable characters. This parameter is used only if *lu_name* is not specified.

If *lu_name* and *lu_alias* are not specified, the command is forwarded to the LU associated with the CP (the default LU).

conv_group_id
Conversation group identifier for the session to be deactivated.

type Type of deactivation. Possible values are:

DEACT_CLEANUP
Deactivate the session immediately, without waiting for a response from the partner LU.

DEACT_NORMAL
Do not deactivate the session until all conversations using the session have ended.

sense_data
If *type* is set to DEACT_CLEANUP, the *sense_data* parameter specifies the sense

data to be used when deactivating the session (specified as a 4-byte hexadecimal number preceded by 0x, for example 0x84000007). Otherwise, this parameter is not used.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

DEACT_CG_INVALID_CGID
The *conv_group_id* parameter did not match any valid conversation group ID.

INVALID_CLEANUP_TYPE
The *type* parameter was not set to a valid value.

INVALID_LU_ALIAS
The *lu_alias* parameter did not match any defined LU alias.

INVALID_LU_NAME
The *lu_name* parameter did not match any defined LU name.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

deactivate_lu_0_to_3

The **deactivate_lu_0_to_3** command requests Communications Server for Linux to deactivate the session for a particular LU for use with 3270 emulation or LUA (an LU of type 0, 1, 2, or 3). Communications Server for Linux deactivates the session by sending a TERM_SELF message to the host for the PLU-SLU session.

This command must be issued to a running node.

Supplied Parameters

Parameter name	Type	Length	Default
[deactivate_lu_0_to_3]			
lu_name	character	8	

Supplied parameters are:

lu_name
LU name of the local LU. This name is a type-A character string.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_LU_NAME
The *lu_name* parameter did not match any defined LU name.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

deactivate_session

The **deactivate_session** command requests Communications Server for Linux to deactivate one or more sessions using a particular local LU/mode/partner LU combination.

To identify a local LU/mode/partner LU combination specify the following:

- Specify the local LU using either the *lu_name* or the *lu_alias* parameter. If neither are specified then the LU associated with the CP (the default LU) is used.
- Specify the mode.
- Specify the remote LU using either the *fqplu_name* or the *plu_alias* parameter.

To deactivate a particular session using the specified local LU/mode/partner LU combination, specify its session identifier. If no session identifier is specified then all sessions using the specified local LU/mode/partner LU are deactivated.

This command must be issued to a running node.

Supplied Parameters

Parameter name	Type	Length	Default
[deactivate_session]			
lu_name	character	8	(null string)
lu_alias	character	8	(null string)
session_id	hex number	8	0x0
plu_alias	character	8	(null string)
mode_name	character	8	
type	constant		DEACT_NORMAL
sense_data	hex number	4	0x0
fqplu_name	character	17	(null string)

Supplied parameters are:

lu_name
: LU name of the local LU. This name is a type-A character string. To indicate that the LU is defined by its LU alias instead of its LU name, do not specify this parameter.

lu_alias
: LU alias of the local LU. This alias is a character string using any locally displayable characters. This parameter is used only if *lu_name* is not specified.

  If *lu_name* and *lu_alias* are not specified, the command is forwarded to the LU associated with the CP (the default LU).

session_id
: An 8-byte identifier of the session to deactivate. If you do not specify this parameter, Communications Server for Linux deactivates all sessions for the LU-MODE-LU combination.

  An error code is not returned when the specified session ID does not match the session ID of an active session (implying that the session has already been deactivated).

plu_alias
: LU alias of the partner LU. This alias is a character string using any locally displayable characters. To indicate that the partner LU is defined by its fully qualified LU name instead of its LU alias, do not specify this parameter.

mode_name
: Name of the mode to be used by the LUs. This name is a type-A character string starting with a letter.

type Type of deactivation. Possible values are:

DEACT_CLEANUP
: Deactivate the session immediately, without waiting for a response from the partner LU.

DEACT_NORMAL
: Do not deactivate the session until all conversations using the session have ended.

sense_data
: If *type* is set to DEACT_CLEANUP, the *sense_data* parameter specifies the sense data to be used when deactivating the session (specified as a 4-byte hexadecimal number preceded by 0x, for example 0x84000007). Otherwise, this parameter is not used.

fqplu_name
Fully qualified name of the partner LU. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character partner LU name.

This parameter is used only if the *plu_alias* parameter is not specified; it is ignored if *plu_alias* is specified.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_CLEANUP_TYPE
The *type* parameter was not set to a valid value.

INVALID_LU_ALIAS
The *lu_alias* parameter did not match any defined LU alias.

INVALID_LU_NAME
The *lu_name* parameter did not match any defined LU name.

INVALID_MODE_NAME
The *mode_name* parameter did not match any defined mode name.

INVALID_PLU_NAME
The *fqplu_name* parameter did not match any defined partner LU name.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

define_adjacent_len_node

define_adjacent_len_node adds entries to the node directory database for an adjacent LEN node and its associated LUs, or adds additional LU entries for a previously-defined LEN node.

This command is equivalent to a series of **define_directory_entry** commands for the LEN node and its associated LUs; it provides a fast method of defining the LEN node's configuration with a single command. To query the directory entries created by this command, use **query_directory_entry**.

If this command is issued to the network node acting as the server for the LEN node, the LEN node's resources are added to the network node's directory database. This means that the network node will respond to network searches for these resources, so that they are accessible to the entire network.

If the command is issued to an end node, the LEN node's resources are accessible only to that end node.

Supplied Parameters

Parameter name	Type	Length	Default
[define_adjacent_len_node]			
cp_name	character	17	
description	character	31	(null string)
lu_name	character	8	
wildcard_lus	constant		NO

(Up to 10 *lu_name* entries may be included.)

Supplied parameters are:

cp_name
: The fully qualified name of the CP in the adjacent LEN end node. This name should match the name the LEN node sends on its XIDs (if the LEN node supports XIDs) and the adjacent CP name specified on the **define_ls** command for the link to the LEN node.

  Specify 3–17 characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character partner LU name.

description
: A text string of 0–31 characters followed by a null character that describes the adjacent node. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_directory_entry** command.

lu_name
: Name of an LU defined on the LEN node. Specify 1–8 type-A characters corresponding to the second part of the fully qualified LU name (the first part of the fully-qualified name is defined by the *cp_name* parameter).

  To define an adjacent node with more than 10 LUs, use multiple **define_adjacent_len_node** commands for the same CP name.

  To define the LU associated with the LEN node's control point (the CP LU or default LU), specify the node's fully qualified CP name in the *cp_name* parameter, and include the 'network name' part of this name (the 8 characters after the period) as one of the LU names.

  You can specify a 'wildcard' LU name to match multiple LU names by specifying only the initial characters of the name. For example, the wildcard LU name `LU` will match `LUNAME` or `LU01` (but will not match `NAMELU`). However, all the LU names specified on a single command must be of the same type (wildcard or explicit), as defined by the *wildcard_lus* parameter. To add both types of LU names for the same adjacent node, use multiple **define_adjacent_len_node** commands.

wildcard_lus
Indicates whether the specified LU names are wildcard entries or explicit LU names. Possible values are:

YES The specified LU names are wildcard entries.

NO The specified LU names are explicit entries.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_CP_NAME
The *cp_name* parameter contained a character that was not valid.

INVALID_LU_NAME
One or more of the specified LU names contained a character that was not valid.

INVALID_WILDCARD_NAME
The *wildcard_lus* parameter was set to YES, but one or more of the specified LU names was already defined on a different parent node.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

INVALID_CP_NAME
The CP name specified by the *cp_name* parameter was already defined in a directory entry and is not a LEN node.

INVALID_LU_NAME
One or more of the LU names specified by the *lu_name* parameter were already defined on a different parent node.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_cn

The **define_cn** command defines a connection network (also known as a virtual routing node or VRN). The command provides the network qualified name of the connection network along with its Transmission Group (TG) characteristics. Also provided is a list of the names of the local ports that can access this connection network.

This command can also be used to add new ports to an existing connection network. (Ports can be removed from an existing connection network by issuing **delete_cn**.)

This command is valid only at a network node or an end node, and not at a LEN node.

Supplied Parameters

Parameter name	Type	Length	Default
[define_cn]			
fqcn_name	character	17	
description	character	32	(null string)
effect_cap	decimal		3686400
connect_cost	decimal		0
byte_cost	decimal		0
security	constant		SEC_NONSECURE
prop_delay	constant		PROP_DELAY_LAN
user_def_parm_1	decimal		0
user_def_parm_2	decimal		0
user_def_parm_3	decimal		0
port_name	character	8	(null string)

(1–239 *port_name* entries can be included)

Supplied parameters are:

fqcn_name
: Fully qualified name of the connection network. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, and followed by a 1–8 character connection network name.

description
: A text string describing the connection network. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_cn** command.

effect_cap
: A decimal value representing the line speed in bits per second.

connect_cost
: Cost per connect time. Valid values are integer values in the range 0–255, where 0 is the lowest cost per connect time and 255 is the highest cost per connect time.

byte_cost
: Cost per byte. Valid values are integer values in the range 0–255, where 0 is the lowest cost per byte and 255 is the highest cost per byte.

security
: Security level of the network. Possible values are:

  SEC_NONSECURE
  : No security.

SEC_PUBLIC_SWITCHED_NETWORK
Data is transmitted over a public switched network.

SEC_UNDERGROUND_CABLE
Data is transmitted over secure underground cable.

SEC_SECURE_CONDUIT
Data is transmitted over a line in a secure conduit that is not guarded.

SEC_GUARDED_CONDUIT
Data is transmitted over a line in a conduit that is protected against physical tapping.

SEC_ENCRYPTED
Data is encrypted before transmission over the line.

SEC_GUARDED_RADIATION
Data is transmitted over a line that is protected against physical and radiation tapping.

prop_delay
Propagation delay. The time that a signal takes to travel the length of the link. Specify one of the following values, according to the type of link:

PROP_DELAY_MINIMUM
Minimum propagation delay.

PROP_DELAY_LAN
Delay is less than .5 microseconds (typical for a LAN).

PROP_DELAY_TELEPHONE
Delay is in the range .5–50 microseconds (typical for a telephone network).

PROP_DELAY_PKT_SWITCHED_NET
Delay is in the range 50–250 microseconds (typical for a packet-switched network).

PROP_DELAY_SATELLITE
Delay is greater than 250 microseconds (typical for a satellite link).

PROP_DELAY_MAXIMUM
Maximum propagation delay.

user_def_parm_1 **through** *user_def_parm_3*
User-defined parameters, which you can use to include other characteristics not covered by the previous parameters. Each of these parameters must be set to a value in the range `0`–`255`.

port_name
Array of port names defined on the connection network. Each port name is an 8-byte string consisting of locally displayable characters that must match the name of a previously-defined port. The port type must be a network type that supports connection networks (Ethernet, Token Ring, Enterprise Extender).

If the *fqcn_name* parameter identifies an existing connection network, the new ports are added to it (without changing any ports already defined on the connection network).

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc

PARAMETER_CHECK

secondary_rc

Possible values are:

DEF_LINK_INVALID_SECURITY
The *security* parameter was not set to one of the valid values.

EXCEEDS_MAX_ALLOWED
Adding the specified ports would exceed the maximum total number of ports on a CN.

INVALID_CN_NAME
The *fqcn_name* parameter contained a character that was not valid or was not in the correct format.

INVALID_PORT_NAME
One or more of the port names specified did not match the name of a defined port.

INVALID_PORT_TYPE
One or more of the specified ports cannot be on a CN because its DLC type is a point-to-point type (such as SDLC) rather than a network type.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc

STATE_CHECK

secondary_rc

PORT_ACTIVE
The port specified by the *port_name* parameter cannot be modified because it is currently active.

Function Not Supported

If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameters:

primary_rc

FUNCTION_NOT_SUPPORTED
The local node is a LEN node. This command is valid only at a network node or an end node.

secondary_rc
(This parameter is not used.)

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_cos

The **define_cos** command adds a class of service (COS) definition or modifies a previously defined COS. The definition specifies TG "rows" and node "rows", which associate a range of node and TG characteristics with weights that are used for route calculation. The lower the weight the more favorable the route.

Supplied Parameters

Parameter name	Type	Length	Default
[define_cos]			
cos_name	character	8	
description	character	31	(null string)
transmission_priority	constant		LOW
{cos_tg_row}			
min_effect_cap	decimal		0
min_connect_cost	decimal		0
min_byte_cost	decimal		0
min_security	constant		SEC_NONSECURE
min_prop_delay	constant		PROP_DELAY_LAN
min_user_def_parm_1	decimal		0
min_user_def_parm_2	decimal		0
min_user_def_parm_3	decimal		0
max_effect_cap	hex	1	0xFF
max_connect_cost	decimal		255
max_byte_cost	decimal		255
max_security	constant		SEC_GUARDED_RADIATION
max_prop_delay	constant		PROP_DELAY_MAXIMUM
max_user_def_parm_1	decimal		0
max_user_def_parm_2	decimal		0
max_user_def_parm_3	decimal		0
weight	decimal		

(Up to eight cos_tg_row subrecords can be included, in ascending order of weight.)

{cos_node_row}			
min_rar	decimal		0
min_status	constant		UNCONGESTED
max_rar	decimal		255
max_status	constant		CONGESTED
weight	decimal		

(Up to eight cos_node_row subrecords can be included, in ascending order of weight.)

Supplied parameters are:

cos_name
Class of service name. This name is a type-A character string starting with a letter.

description
A text string describing the COS. Communications Server for Linux uses

this string for information only. It is stored in the node's configuration file and returned on the **query_cos** command.

transmission_priority
Transmission priority. Possible values are:

LOW The session using this COS is given low priority.

MEDIUM The session using this COS is given medium priority.

HIGH The session using this COS is given high priority.

NETWORK
The session using this COS is given the highest priority.

The following subrecord contains additional parameters:

cos_tg_row
Each TG row contains a set of minimum TG characteristics, a set of maximum TG characteristics, and a weight. When computing the weights for a TG, its characteristics are checked against the minimum and maximum characteristics defined for each TG row. The TG is then assigned the weight of the first TG row which bounds all the TG's characteristics within the limits specified. If the TG characteristics do not satisfy any of the listed TG rows, the TG is considered unsuitable for this COS, and is assigned an infinite weight. The TG rows must be listed in ascending order of weight.

The additional parameters are:

min_effect_cap
Minimum limit for actual bits per second rate (line speed).

min_connect_cost
Minimum limit for cost per connect time. Valid values are integer values in the range `0–255`, where `0` is the lowest cost per connect time and `255` is the highest cost per connect time.

min_byte_cost
Minimum limit for cost per byte. Valid values are integer values in the range `0–255`, where `0` is the lowest cost per byte and `255` is the highest cost per byte.

min_security
Minimum level of security. Possible values are:

SEC_NONSECURE
Data is transmitted over an unsecured network.

SEC_PUBLIC_SWITCHED_NETWORK
Data is transmitted over a public-switched network.

SEC_UNDERGROUND_CABLE
Data is transmitted over secure underground cable.

SEC_SECURE_CONDUIT
Data is transmitted over a line in a secure conduit that is not guarded.

SEC_GUARDED_CONDUIT
Data is transmitted over a line in a conduit that is protected against physical tapping.

SEC_ENCRYPTED
Data is encrypted before transmission over the line.

SEC_GUARDED_RADIATION
Data is transmitted over a line that is protected against physical and radiation tapping.

min_prop_delay
Minimum limits for propagation delay, which is the time that a signal takes to travel the length of the link. Specify one of the following values, according to the type of link:

PROP_DELAY_MINIMUM
Minimum propagation delay.

PROP_DELAY_LAN
Delay is less than .5 microseconds (typical for a LAN).

PROP_DELAY_TELEPHONE
Delay is in the range .5–50 microseconds (typical for a telephone network).

PROP_DELAY_PKT_SWITCHED_NET
Delay is in the range 50–250 microseconds (typical for a packet-switched network).

PROP_DELAY_SATELLITE
Delay is greater than 250 microseconds (typical for a satellite link).

PROP_DELAY_MAXIMUM
Maximum propagation delay.

min_user_def_parm_1 **through** *min_user_def_parm_3*
Minimum limits for user-defined parameters, which you can use to include TG characteristics not covered by previously defined parameters. Each of these parameters must be set to a value in the range 0–255.

max_effect_cap
Maximum limit for actual bits per second rate (line speed). Specify a value in the range 0–603,979,776,000.

max_connect_cost
Maximum limit for cost per connect time. Valid values are integer values in the range 0–255, where 0 is the lowest cost per connect time and 255 is the highest cost per connect time.

max_byte_cost
Maximum limit for cost per byte. Valid values are integer values in the range 0–255, where 0 is the lowest cost per byte and 255 is the highest cost per byte.

max_security
Maximum level of security. Possible values are:

SEC_NONSECURE
Data is transmitted over an unsecured network.

SEC_PUBLIC_SWITCHED_NETWORK
Data is transmitted over a public-switched network.

SEC_UNDERGROUND_CABLE
Data is transmitted over secure underground cable.

SEC_SECURE_CONDUIT
Data is transmitted over a line in a secure conduit that is not guarded.

SEC_GUARDED_CONDUIT
Data is transmitted over a line in a conduit that is protected against physical tapping.

SEC_ENCRYPTED
Data is encrypted before transmission over the line.

SEC_GUARDED_RADIATION
Data is transmitted over a line that is protected against physical and radiation tapping.

max_prop_delay
Maximum limits for propagation delay, which is the time that a signal takes to travel the length of the link. Specify one of the following values, according to the type of link:

PROP_DELAY_MINIMUM
Minimum propagation delay.

PROP_DELAY_LAN
Delay is less than .5 microseconds (typical for a LAN).

PROP_DELAY_TELEPHONE
Delay is in the range .5–50 microseconds (typical for a telephone network).

PROP_DELAY_PKT_SWITCHED_NET
Delay is in the range 50–250 microseconds (typical for a packet-switched network).

PROP_DELAY_SATELLITE
Delay is greater than 250 microseconds (typical for a satellite link).

PROP_DELAY_MAXIMUM
Maximum propagation delay.

max_user_def_parm_1 **through** *max_user_def_parm_3*
Maximum limits for user-defined parameters, which you can use to include TG characteristics not covered by previously described parameters. Each of these parameters must be set to a value in the range 0–255.

weight Weight associated with this TG row.

The following subrecord contains additional parameters:

cos_node_row
Each node row contains a set of minimum node characteristics, a set of maximum node characteristics, and a weight. When computing the weights for a node, its characteristics are checked against the minimum and maximum characteristics defined for each node row. The node is then assigned the weight of the first node row which bounds all the node's characteristics within the limits specified. If the node characteristics do not satisfy any of the listed node rows, the node is considered unsuitable for this COS, and is assigned an infinite weight. The node rows must be listed in ascending order of weight.

The additional parameters are:

min_rar
Specifies the minimum route additional resistance (RAR). Values must be in the range 0–255.

min_status
Specifies the minimum congestion status of the node. Possible values are:

UNCONGESTED
The number of ISR sessions is less than the *isr_sessions_upper_threshold* value in the node's configuration.

CONGESTED
The number of ISR sessions exceeds the *isr_sessions_upper_threshold* value.

max_rar
Specifies the maximum route additional resistance (RAR). Values must be in the range 0–255.

max_status
Specifies the maximum congestion status of the node. Possible values are:

UNCONGESTED
The number of ISR sessions is less than the *isr_sessions_upper_threshold* value in the node's configuration.

CONGESTED
The number of ISR sessions exceeds the *isr_sessions_upper_threshold* value.

weight Weight associated with this node row.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_COS_NAME
The *cos_name* parameter contained a character that was not valid.

INVALID_NUMBER_OF_NODE_ROWS
Too many node rows were specified.

INVALID_NUMBER_OF_TG_ROWS
Too many TG rows were specified.

NODE_ROW_WGT_LESS_THAN_LAST
The node rows were not listed in ascending order of weight.

TG_ROW_WGT_LESS_THAN_LAST
The TG rows were not listed in ascending order of weight.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc

COS_TABLE_FULL
You cannot define a new COS because you would exceed the maximum number of COS definitions allowed for the node (specified by the *cos_cache_size* parameter on the **define_node** command).

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_cpic_side_info

The **define_cpic_side_info** command adds a side information entry to the configuration file or replaces an existing entry. A CPI-C side information entry associates a set of conversation characteristics with a symbolic destination name.

Because CPI-C side information entries are defined as domain resources, this command is not associated with a particular node.

Supplied Parameters

Parameter name	Type	Length	Default
[define_cpic_side_info]			
sym_dest_name	character	8	
description	character	31	(null string)
partner_lu_name	character	17	(null string)
tp_name_type	constant		APPLICATION_TP
tp_name	character	64	(null string)
mode_name	character	8	(null string)
conversation_security_type	constant		NONE
security_user_id	character	10	(null string)
security_password	character	10	(null string)
lu_alias	character	8	(null string)

Supplied parameters are:

sym_dest_name
Symbolic destination name that identifies the side information entry. The name can contain any displayable character.

description
A text string describing the side information entry. Communications Server

for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_cpic_side_info** command.

partner_lu_name
Fully qualified name of the partner LU. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character partner LU name.

tp_name_type
The type of the target TP (the valid characters for a TP name are determined by the TP type). Possible values are:

APPLICATION_TP
Application TP. All characters in the TP name must be valid ASCII characters.

SNA_SERVICE_TP
Service TP. The TP name must be specified as a hex array representing the 8 hexadecimal digits of a 4-character name (for example 0x21F0F0F8). The first character (represented by two hexadecimal digits) must be a hexadecimal value in the range `0x0`–`0x3F`, excluding `0x0E` and `0x0F`; the remaining characters (each represented by two hexadecimal digits) must be valid EBCDIC characters.

tp_name
TP name of the target TP.

mode_name
Name of the mode used to access the target TP.

conversation_security_type
Specifies whether the target TP uses conversation security. Allowed values:

NONE The target TP does not use conversation security.

PROGRAM
The target TP uses conversation security. The *security_user_id* and *security_password* parameters are used to access the target TP.

PROGRAM_STRONG
The target TP uses conversation security. The *security_user_id* and *security_password* parameters are used to access the target TP. However, the local node must not send the password across the network in clear text format. This option can be used only if the remote system supports password substitution.

SAME The target TP uses conversation security and can accept an "already verified" indicator from the local TP. (This value indicates that the local TP was invoked by another TP, and has verified the security user ID and password supplied by this TP.) The *security_user_id* parameter is used to access the target TP; no password is required.

security_user_id
User ID used to access the partner TP. This parameter is not required if the *conversation_security_type* parameter is set to `NONE`.

security_password
Password used to access the partner TP. This parameter is required only if the *conversation_security_type* parameter is set to `PROGRAM` or `PROGRAM_STRONG`.

lu_alias
The alias of the local LU used to communicate with the target TP. This alias is a character string using any locally displayable characters.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

INVALID_SYM_DEST_NAME
The *sym_dest_name* parameter contained a character that was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_default_pu

The **define_default_pu** command defines the PU that is the default for handling Communications Server for Linux management services data. Only one default PU for each node can be defined at any time; a second **define_default_pu** for a different PU name overrides the previous definition.

The **define_default_pu** command enables the user to define, redefine, or modify any field of a default PU. This command also enables the user to delete the default PU, by specifying a null PU name.

If an application issues the MS API verb TRANSFER_MS_DATA without specifying a PU name, the data is routed to the default PU defined for the local node and sent on this PU's session with the host SSCP. For more information about TRANSFER_MS_DATA, refer to the *Communications Server for Linux MS Programmer's Guide*.

Supplied Parameters

Parameter name	Type	Length	Default
[define_default_pu]			
pu_name	character	8	
description	character	31	(null string)

Supplied parameters are:

pu_name
Name of the PU that is to be the default PU; for this definition to have any effect, this must be a PU name already defined as part of an LS definition. This name is a type-A character string starting with a letter.

To delete the default PU, specify all zeros.

description
A text string describing the PU. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_default_pu** command.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_defaults

The **define_defaults** command defines default parameters used by the node.

Supplied Parameters

Parameter name	Type	Length	Default
[define_defaults]			
description	character	31	(null string)
mode_name	character	8	
implicit_plu_forbidden	constant		NO
specific_security_codes	constant		NO
limited_timeout	decimal		0

Supplied parameters are:

description
A text string describing the default parameters. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_defaults** command.

mode_name
Name of the default mode. If an application specifies an unrecognized

mode name when attempting to start a session, the parameters from this mode are used as a default definition for the unrecognized mode.

This mode must be either an SNA-defined mode or a mode defined by a previous **define_mode** command. For more information about SNA-defined modes, refer to the *Communications Server for Linux Administration Guide*. The name is a type-A character string starting with a letter.

implicit_plu_forbidden
Specifies whether Communications Server for Linux puts implicit definitions in place for unknown partner LUs. Possible values are:

YES Communications Server for Linux does not put implicit definitions in place for unknown partner LUs. All partner LUs must be defined explicitly.

NO Communications Server for Linux puts implicit definitions in place for unknown partner LUs.

specific_security_codes
Specifies whether Communications Server for Linux uses specific sense codes on a security authentication or authorization failure. Specific sense codes are only returned to those partner LUs which have reported support for them on the session. Possible values are:

YES Communications Server for Linux uses specific sense codes.

NO Communications Server for Linux does not use specific sense codes.

limited_timeout
Specifies the timeout after which free limited-resource conwinner sessions are deactivated. Specify a value in the range `0`–`65,535` seconds.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
`PARAMETER_CHECK`

secondary_rc

INVALID_MODE_NAME
The *mode_name* parameter did not match any defined mode name.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists additional combinations of primary and secondary return codes that are common to all commands.

define_directory_entry

The **define_directory_entry** command defines a new entry in the node directory database. It provides a network qualified resource name along with a resource type (network node, end node, LU, or Wildcard). This command cannot be used to modify existing entries.

When defining an adjacent node and its LUs, use **define_adjacent_len_node** instead of **define_directory_entry**. This enables you to define the node and its LUs with a single command. The **define_directory_entry** command defines only a single entry, so you need to use multiple commands to define entries for the adjacent node and for its LUs.

You can specify a "wildcard" LU name to match multiple LU names by specifying only the initial characters of the name. For example, the wildcard LU name APPN.LU will match APPN.LUNAME or APPN.LU01 (but will not match APPN.NAMELU).

Supplied Parameters

Parameter name	Type	Length	Default
[define_directory_entry]			
resource_name	character	17	
resource_type	constant		LU_RESOURCE
description	character	31	(null string)
parent_name	character	17	(null string)
parent_type	constant		ENCP_RESOURCE

Supplied parameters are:

resource_name
: Fully qualified name of the resource to be registered. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character resource name.

resource_type
: Specifies the type of the resource to be defined. Possible values are:

  ENCP_RESOURCE
  : End node (EN) or low-entry networking (LEN) node

  NNCP_RESOURCE
  : Network node (NN)

  LU_RESOURCE
  : Logical unit (LU)

  WILDCARD_LU_RESOURCE
  : Wildcard LU name

  The owning control point (CP) on the node must be defined before you can specify an LU or wildcard LU resource type.

description
: A text string describing the directory entry. Communications Server for

Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_directory_entry** and **query_directory_lu** commands.

parent_name
Fully qualified name of the parent resource; for an LU the parent resource is the owning control point and for an end node or LEN node it is the network node server. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character parent name.

Do not specify this parameter in the following cases:

- When registering a network node CP.
- When the command is being issued to an end node or LEN node to define an adjacent LEN node CP with which the local node communicates directly.

parent_type
Specifies the parent type of the resource to be defined. Possible values are:

ENCP_RESOURCE
End node (for an LU resource owned by an end node)

NNCP_RESOURCE
Network node (for an LU resource owned by a network node, or for an EN or LEN resource)

Do not specify this parameter when no parent name is supplied.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_FQ_OWNING_CP_NAME
The *parent_name* parameter did not match the name of a defined parent resource.

INVALID_LU_NAME
The *resource_name* parameter contained a character that was not valid or was not in the correct format.

INVALID_RESOURCE_TYPE
The *resource_type* parameter was not set to a valid value.

INVALID_WILDCARD_NAME
The *resource_type* parameter was set to WILDCARD_LU_RESOURCE but the *resource_name* parameter did not contain a valid wildcard entry.

DUPLICATE
The *resource_name* parameter contained a wildcard entry that has already been defined.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

define_dlur_defaults

The **define_dlur_defaults** command defines a default Dependent LU Server (DLUS) and a backup default DLUS. If a default DLUS or backup default DLUS is already defined, the command overrides the existing definition. DLUR uses the default DLUS name when it initiates SSCP-PU activation for PUs that do not have an explicitly specified associated DLUS. (To define a PU and its associated DLUS, use **define_internal_pu** for a local PU, or **define_*_ls** (for the appropriate link type) for a downstream PU.)

The command can also be used to revoke a default DLUS or backup default DLUS, so that one is not defined.

Supplied Parameters

Parameter name	Type	Length	Default
[define_dlur_defaults]			
description	character	31	(null string)
dlus_name	character	17	
bkup_dlus_name	character	17	(null string)
dlus_retry_timeout	decimal		5
dlus_retry_limit	decimal		3
prefer_active_dlus	constant		

Supplied parameters are:

description
A text string describing the DLUR defaults. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file, but Communications Server for Linux does not make any other use of it.

dlus_name
Name of the DLUS node that is to be the default. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character DLUS node name. To revoke the current default DLUS, so that no default DLUS is defined, do not specify this parameter.

bkup_dlus_name
Name of the DLUS node that is to serve as the backup default. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character backup DLUS node

name. To revoke the current backup default DLUS, so that no backup default DLUS is defined, do not specify this parameter.

dlus_retry_timeout
The interval in seconds between the second and subsequent attempts to contact the DLUS specified by the *dlus_name* and *bkup_dlus_name* parameters. The interval between the first and second attempts is always 1 second.

Specify a value in the range 1– 65,535.

dlus_retry_limit
Retry count for contacting a DLUS. This parameter is used to specify the number of times Communications Server for Linux should retry if it fails to contact the DLUS on the first attempt.

Specify a value in the range 1–65,534, or specify 65,535 to indicate that Communications Server for Linux should retry indefinitely until it contacts the DLUS.

prefer_active_dlus
Specifies how Communications Server for Linux operates when it receives a negative RSP(REQACTPU) from DLUS, or is attempting to reactivate a failed DLUR PU. Possible values are:

YES If either the default primary DLUS or default backup DLUS is active, Communications Server for Linux will attempt to activate or reactivate the PU by using the active DLUS only.

NO Communications Server for Linux will attempt to activate or reactivate the PU by using the standard retry logic.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_DLUS_NAME
The supplied *dlus_name* parameter contained a character that was not valid or was not in the correct format.

INVALID_BKUP_DLUS_NAME
The supplied *bkup_dlus_name* parameter contained a character that was not valid or was not in the correct format.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Function Not Supported

If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameters:

primary_rc

FUNCTION_NOT_SUPPORTED
The local node does not support DLUR; this support is defined by the *dlur_support* parameter on the **define_node** command.

secondary_rc
(This parameter is not used.)

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

define_domain_config_file

The **define_domain_config_file** command specifies an optional comment string to be included in the header of a domain configuration file. If you are creating a domain configuration file using a text editor, this comment string must be the first record in the file.

Supplied Parameters

Parameter name	Type	Length	Default
[define_domain_config_file]			
comment	character	100	(null string)

The supplied parameter is:

comment
An optional comment string containing information about the file. Communications Server for Linux uses the string for information only. It is stored in the node's configuration file and returned on the **query_domain_config_file** command.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_downstream_lu

The **define_downstream_lu** command defines a new downstream LU and maps it to an upstream host LU or LU pool. This enables the downstream LU to access the host computer using the SNA gateway feature of Communications Server for Linux. This command cannot be used to modify an existing downstream LU.

This command can also be used to activate a downstream LU that is already defined (for example, because the downstream workstation has just been activated). To do this, reissue the **define_downstream_lu** command for that LU. Note that all parameters must be the same as in the original definition, because you cannot modify the definition.

define_downstream_lu can also be used to define the downstream LU used by an application that communicates with a Communications Server for Linux Primary RUI application. For more information about Primary RUI, see Communications Server for Linux LUA Programmer's Guide.

Supplied Parameters

Parameter name	Type	Length	Default
[define_downstream_lu]			
dslu_name	character	8	
description	character	31	(null string)
nau_address	decimal		
dspu_name	character	8	
host_lu_name	character	8	
allow_timeout	constant		NO
delayed_logon	constant		NO

Supplied parameters are:

dslu_name
: Name of the downstream LU to be defined. This name is a type-A character string starting with a letter.

description
: A text string describing the downstream LU. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_downstream_lu** command.

nau_address
: Network accessible unit (NAU) address of the downstream LU. This must be in the range 1–255.

dspu_name
: Name of the downstream PU associated with this LU, as specified on **define_*_ls**. This name is a type-A character string starting with a letter.

host_lu_name
: Name of the host LU or host LU pool that the downstream LU uses. This name is an 8-byte type-A character string.

For SNA gateway, the host LU cannot be a dependent LU type 6.2. However, if the downstream LU is LU type 6.2, you can configure the host LU as an LU type 0–3 and specify that the model type for the host LU is unknown.

If the downstream LU is used to communicate with a Communications Server for Linux Primary RUI application instead of a host, set this field to the string `#PRIRUI#`.

allow_timeout
Specifies whether to allow the session between the downstream LU and the upstream LU to timeout if the session is left inactive for the timeout period specified on the upstream LU definition. Possible values are:

YES Allow the session this downstream LU has with the upstream LU to timeout.

NO Do not allow the session this downstream LU has with the upstream LU to timeout.

This field is ignored if the downstream LU is used to communicate with a Communications Server for Linux Primary RUI application instead of a host.

delayed_logon
Specifies whether to use delayed logon with this downstream LU (the upstream LU is not activated until the user requests it). Possible values are:

YES Use delayed logon with this downstream LU; the upstream LU is not activated until the user requests it.

NO Do not use delayed logon with this downstream LU.

This field is ignored if the downstream LU is used to communicate with a Communications Server for Linux Primary RUI application instead of a host.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
`PARAMETER_CHECK`

secondary_rc
Possible values are:

INVALID_DNST_LU_NAME
The supplied *dslu_name* parameter contained a character that was not valid.

INVALID_NAU_ADDRESS
The supplied NAU address was not in the valid range.

INVALID_ALLOW_TIMEOUT
The supplied *allow_timeout* parameter value was not valid.

INVALID_DELAYED_LOGON
The supplied *delayed_logon* parameter value was not valid.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

INVALID_PU_NAME
The specified *dspu_name* parameter value was not valid.

PU_NOT_DEFINED
The specified *dspu_name* parameter did not match any defined PU name.

INVALID_PU_TYPE
The PU specified by the *dspu_name* parameter is not a downstream PU that supports SNA gateway.

LU_ALREADY_DEFINED
An LU with the name specified by the *dslu_name* parameter has already been defined, and cannot be modified using this command.

DSLU_ACTIVE
The LU is already active.

LU_NAU_ADDR_ALREADY_DEFD
An LU with the NAU address specified by the *nau_address* parameter has already been defined.

INVALID_HOST_LU_NAME
The specified *host_lu_name* parameter value was not valid.

LU_NAME_POOL_NAME_CLASH
The specified LU name matches the name of an existing LU pool.

PU_NOT_ACTIVE
The PU specified by the *dspu_name* parameter is not currently active.

LU_ALREADY_ACTIVATING
An LU with the name specified by the *dslu_name* parameter is currently activating.

LU_DEACTIVATING
An LU with the name specified by the *dslu_name* parameter is currently deactivating.

LU_ALREADY_ACTIVE
An LU with the name specified by the *dslu_name* parameter is already active.

Function Not Supported

If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameters:

primary_rc

FUNCTION_NOT_SUPPORTED
The local node does not support SNA gateway; this support is defined by the *pu_conc_support* parameter on the **define_node** command.

secondary_rc
(This parameter is not used.)

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

define_downstream_lu_range

The **define_downstream_lu_range** command defines a range of new downstream LUs and maps them to an upstream host LU or LU pool. This command cannot be used to modify existing downstream LUs.

The supplied parameters include a base name for the new LUs and the range of NAU addresses. The LU base name and NAU addresses are combined to generate the new LU names. For example, a base name of LUNME combined with a NAU address in the range 11–14 results in the following LU names: LUNME011, LUNME012, LUNME013 and LUNME014.

define_downstream_lu_range can also be used to define downstream LUs used by applications that communicate with a Communications Server for Linux Primary RUI application. For more information about Primary RUI, see Communications Server for Linux LUA Programmer's Guide.

Supplied Parameters

Parameter name [define_downstream_lu_range]	Type	Length	Default
dslu_base_name	character	5	
description	character	31	(null string)
min_nau	decimal		1
max_nau	decimal		1
dspu_name	character	8	
host_lu_name	character	8	
allow_timeout	constant		NO
delayed_logon	constant		NO

Supplied parameters are:

dslu_base_name
Base name for the names of the new LUs. This name is a type-A character string of 1–5 characters starting with a letter. Communications Server for Linux generates an LU name for each LU by appending the 3-digit decimal value of the NAU address to the base name.

description
A text string describing the downstream LUs; the same string is used for each LU in the range. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_downstream_lu command**.

min_nau
NAU address of the first LU, in the range 1–255.

max_nau
NAU address of the last LU, in the range 1–255.

dspu_name
Name of the downstream PU (as specified on **define_*_ls**, where * is replaced by the LS type) that the downstream LUs in this range use. The name is a type-A character string starting with a letter.

host_lu_name
Name of the host LU or host LU pool to which the downstream LUs in the given range are mapped. The name is an 8-byte type-A character string.

If the downstream LUs are used to communicate with a Communications Server for Linux Primary RUI application instead of a host, set this field to the string `#PRIRUI#`.

allow_timeout
Specifies whether to allow the sessions this range of downstream LUs have with the upstream LU to timeout if the session is left inactive for the timeout period specified on the upstream LU definition. Possible values are:

YES Allow the sessions this range of downstream LUs have with the upstream LU to timeout.

NO Do not allow the session this range of downstream LUs have with the upstream LU to timeout.

This field is ignored if the downstream LUs are used to communicate with a Communications Server for Linux Primary RUI application instead of a host.

delayed_logon
Specifies whether to use delayed logon with this range of downstream LUs (the upstream LU is not activated until the user requests it). Possible values are:

YES Use delayed logon with this range of downstream LUs; the upstream LU is not activated until the user requests it.

NO Do not use delayed logon with this range of downstream LUs.

This field is ignored if the downstream LUs are used to communicate with a Communications Server for Linux Primary RUI application instead of a host.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_DNST_LU_NAME
The supplied *dslu_base_name* parameter contained a character that was not valid.

INVALID_NAU_ADDRESS
The *min_nau* parameter value, *max_nau* parameter value, or both parameter values were not in the valid range.

INVALID_ALLOW_TIMEOUT
The supplied *allow_timeout* parameter value was not valid.

INVALID_DELAYED_LOGON
The supplied *delayed_logon* parameter value was not valid.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

INVALID_PU_NAME
The specified *dspu_name* parameter value was not valid.

PU_NOT_DEFINED
The specified *dspu_name* parameter did not match any defined PU name.

INVALID_PU_TYPE
The PU specified by the *dspu_name* parameter is not a downstream PU that supports SNA gateway.

LU_ALREADY_DEFINED
An LU has already been defined with a name that matches one of the names in the range. The existing LU cannot be modified using this command.

DSLU_ACTIVE
An LU with a name that matches one of the names in the range is already active. The existing LU cannot be modified using this command.

LU_NAU_ADDR_ALREADY_DEFD
An LU has already been defined with an NAU address that matches one of the addresses in the range.

INVALID_HOST_LU_NAME
The specified *host_lu_name* parameter value was not valid.

LU_NAME_POOL_NAME_CLASH
One of the LU names in the range matches the name of an existing LU pool.

Function Not Supported

If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameters:

primary_rc

FUNCTION_NOT_SUPPORTED
The local node does not support SNA gateway; this support is defined by the *pu_conc_support* parameter on the **define_node** command.

secondary_rc
(This parameter is not used.)

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_dspu_template

The **define_dspu_template** command defines a template for the downstream LUs that use the Communications Server for Linux SNA gateway feature. This template is used to define downstream LUs on a group of downstream workstations when a workstation connects over an implicit link (a link not previously defined)..

define_dspu_template can also be used to define downstream LUs that support applications communicating with a Primary RUI application on the Communications Server for Linux node. For more information about Primary RUI, see Communications Server for Linux LUA Programmer's Guide.

Supplied Parameters

Parameter name	Type	Length	Default
[define_dspu_template]			
template_name	character	8	
description	character	32	(null string)
modify_template	constant		REPLACE_DSPU_TEMPLATE
max_instance	decimal		0
{dslu_template}			
min_nau	decimal		
max_nau	decimal		
host_lu	character	8	
allow_timeout	constant		NO
delayed_logon	constant		NO

Supplied parameters are:

template_name
The name of the template for downstream LUs that are present on a group of downstream workstations.

description
Resource description that is returned on the **query_dspu_template** command.

modify_template
Specifies whether this command should add additional DSLU templates to an existing DSPU template or should replace an existing DSPU template. Possible values are:

MODIFY_DSPU_TEMPLATE
If the named DSPU template does not exist, then it is created. If the named DSPU template does exist, then the DSLU templates supplied to this command are added to the existing DSPU template.

REPLACE_DSPU_TEMPLATE
A new template is created, overwriting any existing definition.

max_instance
The maximum number of instances of the template that can be active concurrently. When the limit is reached, no new instances are created. Specify a value in the range 0–65,535, where 0 indicates no limit.

The subrecord dslu_template contains the following parameters:

min_nau
NAU address of the first downstream PU, in the range 1–255.

max_nau
NAU address of the last downstream PU, in the range 1–255.

host_lu Name of the host LU or host LU pool that the downstream LU uses. This name is an 8-byte type-A character string.

If the downstream LUs are used to communicate with a Communications Server for Linux Primary RUI application instead of a host, set this field to the string #PRIRUI#.

allow_timeout
Specifies whether to timeout host LUs used by the downstream LU if the session is left inactive for the timeout period specified on the host LU definition. Possible values are:

YES Communications Server for Linux is allowed to timeout host LUs used by this downstream LU.

NO Communications Server for Linux is not allowed to timeout host LUs used by this downstream LU.

This field is ignored if the downstream LUs are used to communicate with a Communications Server for Linux Primary RUI application instead of a host.

delayed_logon
Specifies whether to delay connecting the downstream LU to the host LU until the first data is received from the downstream LU. Possible values are:

YES Communications Server for Linux delays connecting the downstream LU to the host LU. A simulated logon screen is sent to the downstream LU.

NO Communications Server for Linux does not delay connecting the downstream LU to the host LU.

This field is ignored if the downstream LUs are used to communicate with a Communications Server for Linux Primary RUI application instead of a host.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
: `PARAMETER_CHECK`

secondary_rc
: Possible values are:

`INVALID_TEMPLATE_NAME`
: The name specified for the *template_name* parameter was not valid.

`INVALID_NAU_RANGE`
: The address specified on the *min_nau* or *max_nau* parameters was not in the valid range.

`CLASHING_NAU_RANGE`
: The range of addresses specified by the *min_nau* parameter through the *max_nau* parameter in a `dslu_template` subrecord clashes with a range specified by another `dslu_template` subrecord in the template named by the *template_name* parameter.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
: `STATE_CHECK`

secondary_rc
: Possible values are:

`INVALID_HOST_LU_NAME`
: The specified *host_lu_name* parameter value was not valid.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_ethernet_dlc

For more information, see "define_tr_dlc, define_ethernet_dlc" on page 207.

define_ethernet_ls

For more information, see "define_tr_ls, define_ethernet_ls" on page 209.

define_ethernet_port

For more information, see “define_tr_port, define_ethernet_port” on page 226.

define_focal_point

The **define_focal_point** command defines a focal point for a particular Management Services category. When a new focal point is defined, Communications Server for Linux attempts to establish an implicit primary focal point relationship with the defined focal point by sending an MS_CAPABILITIES request.

Supplied Parameters

Parameter name	Type	Length	Default
[define_focal_point]			
ms_category	character	8	
fp_fqcp_name	character	17	(null string)
ms_appl_name	character	8	(null string)
description	character	31	(null string)
backup	constant		NO

Supplied parameters are:

ms_category
Management Services category. This category is one of the following:

- One of the category names specified in *SNA Management Services Reference*. Specify the name as a hexadecimal array (for example 0x23F0F3F1).
- A user-defined category name. Specify the name as a type-1134 character string.

fp_fqcp_name
Fully qualified control point name of the focal point to be defined. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character CP name.

To revoke the existing focal point for the specified MS category, do not specify this parameter.

ms_appl_name
Management Services focal point application name. This name is usually a type-1134 character string; alternatively, it can be one of the MS Discipline-Specific Application Program names specified in *SNA Management Services Reference*.

description
A text string describing the focal point. Communications Server for Linux uses this string for information only. It is stored in the node’s configuration file and returned on the **query_focal_point** command.

backup Indicates whether the specified application is the backup focal point or the main focal point for this category. Possible values are:

YES The application is the backup focal point (used only if the main focal point is not available).

NO The application is the main focal point.

Returned Parameters

If the command executes successfully, Communications Server for Linux returns the following parameter:

primary_rc

OK The focal point was defined as requested.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc

PARAMETER_CHECK

secondary_rc

Possible values are:

INVALID_CATEGORY_NAME
The *ms_category* parameter contained a character that was not valid.

INVALID_FP_NAME
The *fp_fqcp_name* or *ms_appl_name* parameter value was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Function Not Supported

If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameters:

primary_rc

FUNCTION_NOT_SUPPORTED
The local node does not support MS network management functions; support is defined by the *mds_supported* parameter on the **define_node** command.

secondary_rc

NO_SECONDARY_RC

Other Conditions

If the command does not execute because other conditions exist, Communications Server for Linux returns the following parameters:

primary_rc

REPLACED
Another **define_focal_point** command was issued to the same node while this command was outstanding, specifying a different focal point for the same MS category. This command was ignored; the node will attempt to contact the focal point specified by the second command.

secondary_rc
NO_SECONDARY_RC

primary_rc
UNSUCCESSFUL

secondary_rc
Possible values are:

IMPLICIT_REQUEST_REJECTED
The specified focal point rejected the request.

IMPLICIT_REQUEST_FAILED
The node could not send the request to the specified focal point; this may be because the specified control point or application was not found.

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_internal_pu

The **define_internal_pu** command defines a PU on the local node that is served by DLUR. (To define a downstream PU served by DLUR or SNA gateway, or to define a local PU that is directly attached to the host, use **define_*_ls** (for the appropriate link type) instead of **define_internal_pu**.)

Supplied Parameters

Parameter name [define_internal_pu]	Type	Length	Default
pu_name	character	8	
description	character	31	(null string)
dlus_name	character	17	(null string)
bkup_dlus_name	character	17	(null string)
pu_id	hex array	4	
initially_active	constant		NO
dlus_retry_timeout	decimal		(0)
dlus_retry_limit	decimal		(0)
conventional_lu_compression	constant		NO
dddlu_offline_supported	constant		NO

Supplied parameters are:

pu_name
Name of the internal PU to be defined. This name is a type-A character string starting with a letter.

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

description
A text string describing the internal PU. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_dlur_pu** and **query_pu** commands.

dlus_name
Name of the DLUS node that DLUR will use when it initiates SSCP-PU

activation. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character DLUS node name.

To indicate that DLUR should use the global default DLUS, do not specify this parameter. In this case, you must also use **define_dlur_defaults** to define a global default DLUS.

bkup_dlus_name
Name of the DLUS node that will serve as the backup DLUS for this PU. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character backup DLUS name.

To indicate that DLUR should use the global backup default DLUS, do not specify this parameter. In this case, you must also use **define_dlur_defaults** to define a global backup default DLUS.

pu_id PU identifier. This ID is a 4-byte hexadecimal string, consisting of a block number (three hexadecimal digits) and a node number (five hexadecimal digits). The PU ID must match the *pu_id* configured at the host.

initially_active
Specifies whether this internal PU is automatically started when the node is started. Possible values are:

YES The PU is automatically started when the node is started.

NO The PU is not automatically started; it must be manually started.

dlus_retry_timeout
The interval in seconds between the second and subsequent attempts to contact the DLUS specified by the *dlus_name* and *bkup_dlus_name* parameters. The interval between the first and second attempts is always 1 second.

Specify a value in the range 0– 65,535. If 0 (zero) is specified, then the defaults specified using the **define_dlur_defaults** command are used.

dlus_retry_limit
Retry count for contacting a DLUS. This parameter is used to specify the number of times Communications Server for Linux should retry if it fails to contact the DLUS on the first attempt.

Specify a value in the range 1–65,534, or specify 65,535 to indicate that Communications Server for Linux should retry indefinitely until it contacts the DLUS.

conventional_lu_compression
Specifies whether data compression is requested for LU 0–3 sessions using this PU. Possible values are:

YES Data compression should be used for LU 0–3 sessions using this PU if the host requests it.

NO Data compression should not be used for LU 0–3 sessions using this PU.

dddlu_offline_supported
Specifies whether the local PU should send NMVT (power off) messages to the host. If the host system supports DDDLU (Dynamic Definition of Dependent LUs), Communications Server for Linux sends NMVT (power

off) to the host when it has finished using a dynamically defined LU. This allows the host to save resources by removing the definition when it is no longer required.

Possible values are:

YES The local PU sends NMVT (power off) messages to the host.

NO The local PU does not send NMVT (power off) messages to the host.

If the host supports DDDLU but does not support the NMVT (power off) message, this parameter must be set to NO.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_PU_NAME
The *pu_name* parameter contained a character that was not valid.

INVALID_PU_ID
The *pu_id* parameter contained a character that was not valid.

INVALID_DLUS_NAME
The *dlus_name* parameter contained a character that was not valid or was not in the correct format.

INVALID_BKUP_DLUS_NAME
The *bkup_dlus_name* parameter contained a character that was not valid or was not in the correct format.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc

PU_ALREADY_DEFINED
A PU with the specified name has already been defined.

Function Not Supported

If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameters:

primary_rc

FUNCTION_NOT_SUPPORTED
The local node does not support DLUR; this support is defined by the *dlur_support* parameter on the **define_node** command.

secondary_rc
(This parameter is not used.)

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_ip_dlc

The **define_ip_dlc** command defines a new DLC for use with Enterprise Extender (HPR/IP). The command can also be used to modify an existing DLC, if the DLC is not currently active.

Supplied Parameters

Parameter name [define_ip_dlc]	Type	Length	Default
dlc_name	character	8	
description	character	31	(null string)
initially_active	constant		YES
udp_port_llc	decimal		12000
udp_port_network	decimal		12001
udp_port_high	decimal		12002
udp_port_medium	decimal		12003
udp_port_low	decimal		12004
ip_precedence_llc	decimal		6
ip_precedence_network	decimal		6
ip_precedence_high	decimal		4
ip_precedence_medium	decimal		2
ip_precedence_low	decimal		1
no_dns_lookup	constant		NO

Supplied parameters are:

dlc_name
Name of the DLC. This name is a character string using any locally displayable characters.

description
A text string describing the DLC. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_dlc** command.

initially_active
Specifies whether this DLC is automatically started when the node is started. Possible values are:

YES The DLC is automatically started when the node is started.

NO The DLC is automatically started only if a port or LS that uses the DLC is defined as initially active; otherwise, it must be manually started.

udp_port_llc
UDP port number used for LLC commands.

udp_port_network
UDP port number used for network priority traffic.

udp_port_high
UDP port number used for high priority traffic.

udp_port_medium
UDP port number used for medium priority traffic.

udp_port_low
UDP port number used for low priority traffic.

ip_precedence_llc
IP precedence value used for LLC commands, in the range `0` (minimum)—7 (maximum).

ip_precedence_network
IP precedence value used for network priority traffic, in the range `0` (minimum)—7 (maximum).

ip_precedence_high
IP precedence value used for high priority traffic, in the range `0` (minimum)—7 (maximum).

ip_precedence_medium
IP precedence value used for medium priority traffic, in the range `0` (minimum)—7 (maximum).

ip_precedence_low
IP precedence value used for low priority traffic, in the range `0` (minimum)—7 (maximum).

no_dns_lookup
Specifies whether remote host IP addresses require lookup at a Domain Name Server. Possible values are:

YES Do not attempt to look up the hostname from the remote IP address when receiving an incoming IP connection.

Use this option when the remote IP address cannot be resolved. In this case, the incoming connection can be matched to a configured LS only if the LS is configured to use an explicit IP address (either IPv4 or IPv6) rather than a hostname.

NO The remote host IP address on link stations defined for this DLC can be specified as a numeric address (either IPv4 or IPv6), as a name (such as `newbox.this.co.uk`), or as an alias (such as `newbox`). The node performs a Domain Name Server lookup to determine the remote host name on all incoming calls where necessary.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

INVALID_DLC_NAME
The supplied *dlc_name* parameter contained a character that was not valid.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc

DLC_ACTIVE
A parameter cannot be modified because the DLC is currently active.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_ip_ls

The **define_ip_ls** command is used to define a new link station (LS) for Enterprise Extender (HPR/IP), or modify an existing one. Before issuing this command, you must define the port that this LS uses.

You cannot use this command to modify the port used by an existing LS; the name of the port specified in the *port_name* parameter for this command must match the previous definition of the LS. The LS can be modified only if it is not started.

Supplied Parameters

Parameter name	Type	Length	Default
[define_ip_ls]			
ls_name	character	8	
description	character	31	(null string)
port_name	character	8	
adj_cp_name	character	17	(null string)
adj_cp_type	constant		LEARN_NODE
lsap_address	decimal		4
auto_act_supp	constant		NO
tg_number	decimal		0
limited_resource	constant		NO
disable_remote_act	constant		NO
link_deact_timer	decimal		30
default_nn_server	constant		NO
ls_attributes	constant		SNA
adj_node_id	hex array	4	
local_node_id	hex array	4	
cp_cp_sess_support	constant		YES
use_default_tg_chars	constant		YES
effect_cap	decimal		865075200 (Linux on System z) 157286400 (other Linux variants)
connect_cost	decimal		0
byte_cost	decimal		0
security	constant		SEC_NONSECURE
prop_delay	constant		PROP_DELAY_MINIMUM
user_def_parm_1	decimal		0
user_def_parm_2	decimal		0
user_def_parm_3	decimal		0
target_pacing_count	decimal		7
max_send_btu_size	decimal		1461
conventional_lu_compression	constant		NO
initially_active	constant		NO
react_timer	decimal		30
react_timer_retry	decimal		65535
restart_on_normal_deact	constant		NO
max_ifrm_rcvd	decimal		0
branch_link_type	constant		UPLINK (used only if this node is BrNN)
adj_brnn_cp_support	constant		ALLOWED (used only if this node is BrNN)
ip_ack_timeout	decimal		2000
ip_max_retry	decimal		10
liveness_timeout	decimal		10000
short_hold_mode	constant		NO
ip_version	constant		IPV4
remote_ip_host	character	255	

Supplied parameters are:

ls_name
: Name of the link station to be defined.

description
: A text string describing the LS. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_ls**, **query_pu**, and **query_downstream_pu** commands.

port_name
: Name of the port associated with this link station. This name must match the name of a defined port.

adj_cp_name
Fully qualified name of the adjacent CP for this LS. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character CP name. This parameter is used in the following ways:

- If the *adj_cp_type* parameter is set to NETWORK_NODE or END_NODE, and preassigned TG numbers are being used, set this parameter to the CP name defined at the adjacent node; if the adjacent node sends a CP name during XID exchange, it will be checked against this value.
- If *adj_cp_type* is set to any other value, or if preassigned TG numbers are not being used, there is no need to specify this parameter; Communications Server for Linux checks the CP name only if one is specified.

adj_cp_type
Adjacent node type.

If preassigned TG numbers are not being used, this parameter is usually set to LEARN_NODE, indicating that the node type is unknown; Communications Server for Linux will determine the type during XID exchange.

If preassigned TG numbers are being used, you must specify the node type explicitly. You can also specify the type as an additional security check if preassigned TG numbers are not being used. In this case, Communications Server for Linux will reject a connection attempt from the adjacent node if its node type does not match the one specified for this parameter. Possible values are:

LEARN_NODE
The adjacent node type is unknown; Communications Server for Linux will determine the type during XID exchange.

END_NODE
The adjacent node is an End Node, a Branch Network Node acting as an End Node from the local node's perspective, or a LEN node that includes the Network Name CV in its XID3.

NETWORK_NODE
The adjacent node is a Network Node, or a Branch Network Node acting as a Network Node from the local node's perspective.

lsap_address
Link Service Access Point address used by the local link station. This must match the address used by the remote station. Specify a multiple of 4. The usual value is 4, but VTAM® may use 8 in some circumstances.

If you need to use two or more ports with different LSAP addresses on the same TCP/IP interface, you will need to create two or more Enterprise Extender DLCs, and then create a separate Enterprise Extender port for each DLC with the same *if_name* but a different LSAP address.

auto_act_supp
Specifies whether the link can be automatically activated when required by a session. Possible values are:

YES The link can be automatically activated.

The reactivation timer parameters are ignored. If the LS fails, Communications Server for Linux does not attempt to reactivate it until a dependent LU application that was using the session

attempts to restart the session; an LS used by independent LUs is not reactivated by Communications Server for Linux and must be manually restarted.

The following restrictions also apply:

- If the link is to an APPN node, the LS must have a preassigned TG number defined (see the *tg_number* parameter), and *cp_cp_sess_support* must be set to NO.
- If either the local node or the adjacent node is an end node, the LS must also be defined to automatically activate at the adjacent node.

NO The link cannot be automatically activated.

tg_number

Preassigned TG number. This parameter is used only if the adjacent node is an APPN node (*adj_cp_type* is either NETWORK_NODE or END_NODE); it is ignored otherwise.

This TG number is used to represent the link when the link is activated. The node does not accept any other number from the adjacent node during activation of this link; if the adjacent node is using preassigned TG numbers, the same TG number must be defined by the adjacent node on the adjacent link station.

If the local node is a LEN node, or if the adjacent node is a LEN node and the link is to be automatically activated, set the TG number to 1. Otherwise, specify a number in the range 1–20, or 0 (zero) to indicate that the TG number is not preassigned and is negotiated when the link is activated.

If a preassigned TG number is defined, the *adj_cp_name* parameter must also be defined, and the *adj_cp_type* parameter must be set to either END_NODE or NETWORK_NODE.

limited_resource

Specifies whether this link station is a limited resource and is automatically deactivated when there are no sessions using the link. Link stations on a nonswitched port cannot be configured as limited resource. Possible values are:

NO The link is not a limited resource and is not automatically deactivated.

NO_SESSIONS
The link is a limited resource and is automatically deactivated when no active sessions are using it.

INACTIVITY
The link is a limited resource and is automatically deactivated when no active sessions are using it, or when no data has flowed on the link for the time period specified by the *link_deact_timer* parameter.

- If no SSCP-PU session is active across the link, the node deactivates the link immediately.
- If an SSCP-PU session is active but no traffic has flowed for the specified time period, the node sends REQDISCONT(normal) to the host. The host is then responsible for deactivating all LUs and the PU, at which time the local node will deactivate the link. However, the host may not deactivate LUs with active PLU-SLU

sessions; in this case, the link remains active until all these sessions are deactivated (for example by the user logging out). This behavior can be changed by using options in the *ptf* parameter of the **start_node** command.

A limited resource link station can be configured for CP-CP session support by setting this parameter to NO_SESSIONS and *cp_cp_sess_support* to YES. In this case, if CP-CP sessions are brought up over the link, Communications Server for Linux does not treat the link as a limited resource and therefore does not deactivate it.

disable_remote_act
Specifies whether to prevent activation of the LS by the remote node. Possible values are:

YES The LS can be activated only by the local node; if the remote node attempts to activate it, Communications Server for Linux will reject the attempt.

NO The LS can be activated by the remote node.

link_deact_timer
Limited resource link deactivation timer, in seconds. A limited resource link is automatically deactivated if no data flows over the link for the time specified by this parameter. This parameter is not used if *limited_resource* is set to any value other than INACTIVITY.

The minimum value is 5; values in the range 1–4 will be interpreted as 5.

The value 0 (zero) indicates that the default deactivation timer value of 30 is used.

default_nn_server
For an end node this parameter specifies whether the link station being defined supports CP-CP sessions to a network node that can act as the local node's network node server. When the local node has no CP-CP sessions to a network node server and needs to establish them, the local node checks this parameter on its defined link stations to find a suitable LS to activate. This enables you to specify which adjacent NNs are suitable to act as the NN server (for example, to avoid using NNs that are accessed by expensive or slow links).

Possible values are:

YES This link supports CP-CP sessions to a network node that can act as the local node's NN server; the local node can automatically activate this link if it needs to contact an NN server. The *cp_cp_sess_support* parameter must be set to YES.

NO This link does not support CP-CP sessions to a network node that can act as the local node's NN server; the local node cannot automatically activate this link if it needs to contact an NN server.

If the local node is not an end node, this parameter is ignored.

ls_attributes
Attributes of the remote system with which Communications Server for Linux is communicating.

Specify SNA unless you are communicating with a host of one of the other following types. Possible values are:

SNA Standard SNA host

FNA Fujitsu Network Architecture (VTAM-F) host

HNA Hitachi Network Architecture host

SUPPRESS_CP_NAME
Suppress the CP name associated with the remote node. Use a + character to combine this value with SNA, FNA, or HNA.

If *adj_cp_type* is set to BACK_LEVEL_LEN_NODE, and the remote LEN node associated with this LS cannot accept the Network Name CV in the format 3 XID it receives, use a + character to combine the value SNA, FNA, or HNA with SUPPRESS_CP_NAME (for example, SNA+SUPPRESS_CP_NAME).

If *adj_cp_type* is set to any other value, the SUPPRESS_CP_NAME option is ignored.

adj_node_id
Node ID of adjacent node. This ID is a 4-byte hexadecimal string, consisting of a block number (three hexadecimal digits) and a node number (five hexadecimal digits). To disable node ID checking, do not specify this parameter. If this link station is defined on a switched port, the *adj_node_id* must be unique, and there may only be one null *adj_node_id* on each switched port.

local_node_id
Node ID sent in XIDs on this LS. This ID is a 4-byte hexadecimal string, consisting of a block number (three hexadecimal digits) and a node number (five hexadecimal digits). To use the node ID specified in the *node_id* parameter on the **define_node** command, do not specify this parameter.

cp_cp_sess_support
Specifies whether CP-CP sessions are supported. This parameter is valid only if the adjacent node is an end node or a network node (*adj_cp_type* is NETWORK_NODE, END_NODE, or LEARN_NODE); it is ignored otherwise. If both the local node and the adjacent node are network nodes, this parameter should be set to YES in order to use APPN functions between these nodes.

Possible values are:

YES CP-CP sessions are supported.

NO CP-CP sessions are not supported.

use_default_tg_chars
Specifies whether to use the default TG characteristics supplied on **define_ip_port**. The TG characteristics apply only if the link is to an APPN node. If the link is not to an APPN node, the *use_default_tg_chars* through *user_def_parm_3* parameters are ignored. Possible values are:

YES Use the default TG characteristics; ignore the *effect_cap* through *user_def_parm_3* parameters on this command.

NO Use the *effect_cap* through *user_def_parm_3* parameters on this command.

effect_cap
A decimal value representing the line speed in bits per second.

Ensure that you set this parameter to the true 'effective capacity' of the link, including any step-downs or bottlenecks in the path, and not just to the theoretical capacity of the adapter used by the link.

connect_cost
Cost per connect time. Valid values are integer values in the range 0–255, where 0 is the lowest cost per connect time and 255 is the highest cost per connect time.

byte_cost
Cost per byte. Valid values are integer values in the range 0–255, where 0 is the lowest cost per byte and 255 is the highest cost per byte.

security
Security level of the network. Possible values are:

SEC_NONSECURE
No security.

SEC_PUBLIC_SWITCHED_NETWORK
Data is transmitted over a public switched network.

SEC_UNDERGROUND_CABLE
Data is transmitted over secure underground cable.

SEC_SECURE_CONDUIT
Data is transmitted over a line in a secure conduit that is not guarded.

SEC_GUARDED_CONDUIT
Data is transmitted over a line in a conduit that is protected against physical tapping.

SEC_ENCRYPTED
Data is encrypted before transmission over the line.

SEC_GUARDED_RADIATION
Data is transmitted over a line that is protected against physical and radiation tapping.

prop_delay
Propagation delay (the time that a signal takes to travel the length of the link). Specify one of the following values, according to the type of link:

PROP_DELAY_MINIMUM
Minimum propagation delay.

PROP_DELAY_LAN
Delay is less than .5 microseconds (typical for a LAN).

PROP_DELAY_TELEPHONE
Delay is in the range .5–50 microseconds (typical for a telephone network).

PROP_DELAY_PKT_SWITCHED_NET
Delay is in the range 50–250 microseconds (typical for a packet-switched network).

PROP_DELAY_SATELLITE
Delay is greater than 250 microseconds (typical for a satellite link).

PROP_DELAY_MAXIMUM
Maximum propagation delay.

user_def_parm_1 **through** *user_def_parm_3*
User-defined parameters, that you can use to include other TG characteristics not covered by the previous parameters. Each of these parameters must be set to a value in the range 0–255.

target_pacing_count
Indicates the desired pacing window size. Specify a value in the range 1–32,767.

max_send_btu_size
Maximum BTU size that can be sent from this link station. This value is used to negotiate the maximum BTU size that a pair of link stations can use to communicate with each other. The value includes the length of the TH and RH (total 9 bytes) as well as the RU. Specify a value in the range 265–65,535.

conventional_lu_compression
Specifies whether data compression is requested for LU 0–3 sessions on this link. This parameter is used only if this link carries LU 0–3 traffic; it does not apply to LU 6.2 sessions. Possible values are:

YES Data compression should be used for LU 0–3 sessions on this link if the host requests it.

NO Data compression should not be used for LU 0–3 sessions on this link.

initially_active
Specifies whether this LS is automatically started when the node is started. Possible values are:

YES The LS is automatically started when the node is started.

NO The LS is not automatically started; it must be manually started.

If the LS is a leased link, you are recommended to set this parameter to YES to ensure that the link is always available.

react_timer
Reactivation timer for reactivating a failed LS. If the *react_timer_retry* parameter is a nonzero value (to specify that Communications Server for Linux should retry activating the LS if it fails), this parameter specifies the time in seconds between retries. When the LS fails, or when an attempt to reactivate it fails, Communications Server for Linux waits for the specified time before retrying the activation. If *react_timer_retry* is 0 (zero), this parameter is ignored.

react_timer_retry
Retry count for reactivating a failed LS. This parameter is used to specify whether Communications Server for Linux should attempt to reactivate the LS if it fails while in use (or if an attempt to start the LS fails).

Specify 0 (zero) to indicate that Communications Server for Linux should not attempt to reactivate the LS, or specify the number of retries to be made. A value of 65,535 indicates that Communications Server for Linux should retry indefinitely until the LS is reactivated.

Communications Server for Linux waits for the time specified by the *react_timer* parameter between successive retries. If the LS is not successfully reactivated by the end of the retry count, or if **stop_ls** is issued while Communications Server for Linux is retrying the activation, no further retries are made; the LS remains inactive unless **start_ls** is issued for it.

If the *auto_act_supp* parameter is set to YES, the *react_timer* and *react_timer_retry* parameters are ignored; if the link fails, Communications

Server for Linux does not attempt to reactivate it until the user application that was using the session attempts to restart the session.

If the LS is a leased link, you are recommended to set this parameter to a non-zero value to ensure that the link is always available.

restart_on_normal_deact
Specifies whether Communications Server for Linux should attempt to reactivate the LS if it is deactivated normally by the remote system. Possible values are:

YES If the remote system deactivates the LS normally, Communications Server for Linux attempts to reactivate it, using the same retry timer and count values as for reactivating a failed LS (the *react_timer* and *react_timer_retry* parameters above).

NO If the remote system deactivates the LS normally, Communications Server for Linux does not attempt to reactivate it.

If the LS is a host link (specified by the *adj_cp_type* parameter), or is automatically started when the node is started (the *initially_active* parameter is set to YES), this parameter is ignored; Communications Server for Linux always attempts to reactivate the LS if it is deactivated normally by the remote system (unless *react_timer_retry* is zero).

max_ifrm_rcvd
Maximum number of I-frames that can be received by the local link stations before an acknowledgment is sent. Specify a value in the range 1–127.

branch_link_type
This parameter applies only if the local node is a Branch Network Node; it is not used if the local node is any other type.

If the parameter *adj_cp_type* is set to NETWORK_NODE, END_NODE, APPN_NODE, or BACK_LEVEL_LEN_NODE, this parameter defines whether the link is an uplink or a downlink. Possible values are:

UPLINK The link is an uplink.

DOWNLINK
The link is a downlink.

If *adj_cp_type* is set to NETWORK_NODE, this parameter must be set to UPLINK.

adj_brnn_cp_support
This parameter applies only if the local node is a Branch Network Node and the adjacent node is a network node (the parameter *adj_cp_type* is set to NETWORK_NODE, or it is set to APPN_NODE and the node type discovered during XID exchange is network node). It is not used if the local and remote nodes are any other type.

This parameter defines whether the adjacent node can be a Branch Network Node that is acting as a Network Node from the point of view of the local node. Possible values are:

ALLOWED
The adjacent node is allowed (but not required) to be a Branch Network Node.

REQUIRED
The adjacent node must be a Branch Network Node.

PROHIBITED
The adjacent node must not be a Branch Network Node.

If *adj_cp_type* is set to NETWORK_NODE and *auto_act_supp* is set to YES, this parameter must be set to REQUIRED or PROHIBITED.

ip_ack_timeout
Duration for the acknowledgment timer (sometimes called the T1 timer): the time in milliseconds within which a response must be received for a command frame sent to the adjacent link station. If the response is not received within this time, a duplicate frame is sent.

A lower value for this parameter means that lost packets will be detected quickly, but may increase network traffic.

Specify a value in the range 0–65535. This parameter should be set to a value greater than twice the expected network latency. A typical value is 2000 milliseconds.

ip_max_retry
The maximum number of times that the local station will retry sending a command frame. If this retry count is exceeded without receiving a response, the link is considered to have failed.

A lower value for this parameter means that link failures will be detected quickly, but may cause unnecessary reporting of link failures if a few packets are lost.

Specify a value in the range 0–255. A typical value is 10 retries.

liveness_timeout
Duration for the liveness timer (sometimes called the TL timer): the time in milliseconds for which the link will be held active if there is no evidence that the remote station is still active.

A lower value for this parameter means that link failures will be detected quickly, but may increase network traffic on idle active links.

Specify a value in the range 1–65535 milliseconds. A typical value is 10000 (10 seconds).

short_hold_mode
Specifies whether the liveness protocol runs only if there has been no evidence that the remote system is still active since data was last transmitted (YES or NO).

Setting this parameter to YES allows links to stay active and idle without unnecessary data traffic, but means that link failures are not detected until the local station attempts to send data. In general this parameter should be set to NO.

ip_version
Specifies whether the following field represents an IPv4 or IPv6 address. This must match the *ip_version* parameter for the port that this LS uses (identified by the *port_name*). Possible values:

IPV4 The *remote_ip_host* field specifies an IPv4 address, or a hostname or alias that resolves to an IPv4 address.

IPV6 The *remote_ip_host* field specifies an IPv6 address, or a hostname or alias that resolves to an IPv6 address.

remote_ip_host
Remote host name of the destination node for this link. This can be any of the following; the *ip_version* parameter determines whether it is an IPv4 or IPv6 address.

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

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

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
`PARAMETER_CHECK`

secondary_rc
Possible values are:

CANT_MODIFY_PORT_NAME
The *ls_name* parameter matched the name of an existing LS, but the *port_name* parameter did not match the existing definition. You cannot modify the port name when changing the definition of an existing LS.

DEF_LINK_INVALID_SECURITY
The *security* parameter was not set to a valid value.

INVALID_AUTO_ACT_SUPP
The *auto_act_supp* parameter was not set to a valid value or was set to `YES` when *cp_cp_sess_support* was also set to `YES`.

INVALID_CP_NAME
The *adj_cp_name* parameter contained a character that was either not valid, not in the correct format, or not specified when required.

INVALID_LIMITED_RESOURCE
The *limited_resource* parameter was not set to a valid value.

INVALID_LINK_NAME
The *ls_name* parameter contained a character that was not valid.

INVALID_NODE_TYPE
The *adj_cp_type* parameter was not set to a valid value.

INVALID_PORT_NAME
The *port_name* parameter did not match the name of any defined port.

INVALID_TARGET_PACING_CNT
The *target_pacing_count* parameter was not set to a valid value.

HPR_NOT_SUPPORTED
A reserved parameter was set to a nonzero value.

INVALID_TG_NUMBER
The TG number supplied was not in the valid range.

MISSING_CP_NAME
A TG number was defined, but no CP name was supplied.

MISSING_CP_TYPE
A TG number was defined, but no CP type was supplied.

MISSING_TG_NUMBER
The link was defined as automatically activated, but no TG number was supplied.

UNKNOWN_IP_HOST
The string specified for the *remote_hostname* parameter could not be resolved to a valid IP address.

INVALID_IP_VERSION
The value specified in the *ip_version* parameter did not match the value specified for the owning IP port.

INVALID_BRANCH_LINK_TYPE
The *branch_link_type* parameter was not set to a valid value.

INVALID_BRNN_SUPPORT
The *adj_brnn_cp_support* parameter was not set to a valid value.

BRNN_SUPPORT_MISSING
The *adj_brnn_cp_support* parameter was set to `ALLOWED`; this value is not valid because the adjacent node is a Network Node and *auto_act_supp* is set to `YES`.

INVALID_UPLINK
The *branch_link_type* parameter was set to `UPLINK`, but the definition of an existing LS between the local and adjacent nodes specifies that it is a downlink. The branch link type must be the same for all LSs between the same two nodes.

INVALID_DOWNLINK
The *branch_link_type* parameter was set to `DOWNLINK`, but the definition of an existing LS between the local and adjacent nodes specifies that it is an uplink. The branch link type must be the same for all LSs between the same two nodes.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
`STATE_CHECK`

secondary_rc
Possible values are:

DUPLICATE_CP_NAME
The CP name specified in the *adj_cp_name* parameter has already been defined.

DUPLICATE_DEST_ADDR
The destination address specified in the *address* parameter has already been defined.

INVALID_LINK_NAME
The link station value specified in the *ls_name* parameter was not valid.

INVALID_NUM_LS_SPECIFIED
The number of link stations specified was not valid.

LOCAL_CP_NAME
The value specified in the *adj_cp_name* parameter was the same as the local CP name.

LS_ACTIVE
The link station specified in the *ls_name* parameter is currently active.

DUPLICATE_TG_NUMBER
The TG number specified in the *tg_number* parameter has already been defined.

TG_NUMBER_IN_USE
The TG number specified in the *tg_number* parameter is in use by another link station.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_ip_port

The **define_ip_port** command is used to define a new port for use with Enterprise Extender (HPR/IP), or to modify an existing port. Before issuing this command, you must define the DLC that this port uses.

You can modify an existing port only if it is not started. You cannot change the DLC used by an existing port; the *dlc_name* specified when modifying an existing port must match the DLC that was specified on the initial definition of the port.

For information about defining a port that will accept incoming calls, see "Incoming Calls" on page 84.

Supplied Parameters

Parameter name	Type	Length	Default
[define_ip_port]			
port_name	character	8	
description	character	31	(null string)
dlc_name	character	8	
max_rcv_btu_size	decimal		1461
tot_link_act_lim	decimal		1
inb_link_act_lim	decimal		0
out_link_act_lim	decimal		1
nonact_xid_exchange_limit	decimal		10
max_ifrm_rcvd	decimal		7
target_pacing_count	decimal		7
max_send_btu_size	decimal		1461
ip_version	constant		IPV4
implicit_cp_cp_sess_support	constant		YES
implicit_limited_resource	constant		NO
implicit_deact_timer	decimal		30
implicit_uplink_to_en	constant		NO
effect_cap	decimal		865075200 (Linux on System z) 157286400 (other Linux variants)
connect_cost	decimal		0
byte_cost	decimal		0
security	constant		SEC_NONSECURE
prop_delay	constant		PROP_DELAY_MINIMUM
user_def_parm_1	decimal		0
user_def_parm_2	decimal		0
user_def_parm_3	decimal		0
initially_active	constant		YES
implicit_ls_limit	decimal		0
ip_ack_timeout	decimal		2000
ip_max_retry	decimal		10
liveness_timeout	decimal		10000
short_hold_mode	constant		NO
local_ip_interface	character	45	

Supplied parameters are:

port_name
: Name of the port to be defined. This name is a character string using any locally displayable characters.

description
: A text string describing the port. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_port** command.

dlc_name
: Name of the associated DLC. This name is a character string using any locally displayable characters. The specified DLC must have already been defined.

max_rcv_btu_size
: Maximum BTU size that can be received. This value includes the length of the TH and RH (total 9 bytes), as well as the RU. Specify a value in the range 265–65,535.

tot_link_act_lim
: Total link activation limit (the maximum number of links that can be active at any time using this port).

inb_link_act_lim
: Inbound link activation limit (the number of links reserved for inbound

activation). The sum of *inb_link_act_lim* and *out_link_act_lim* must not exceed *tot_link_act_lim*; the difference between *inb_link_act_lim* and *tot_link_act_lim* defines the maximum number of links that can be activated outbound at any time.

out_link_act_lim
Outbound link activation limit (the number of links reserved for outbound activation). The sum of *inb_link_act_lim* and *out_link_act_lim* must not exceed *tot_link_act_lim*; the difference between *out_link_act_lim* and *tot_link_act_lim* defines the maximum number of links that can be activated inbound at any time.

act_xid_exchange_limit
Activation XID exchange limit. Specify a value in the range 1–65,535.

nonact_xid_exchange_limit
Nonactivation XID exchange limit. Specify a value in the range 1–65,535.

max_ifrm_rcvd
Maximum number of I-frames that can be received by the local link stations before an acknowledgment is sent. Specify a value in the range 1–127.

target_pacing_count
Indicates the desired pacing window size. Specify a value in the range 1–32,767.

max_send_btu_size
Maximum BTU size that can be sent from this port. This value is used to negotiate the maximum BTU size that a pair of link stations can use to communicate with each other. The value includes the length of the TH and RH (total 9 bytes), as well as the RU. Specify a value in the range 265–65,535.

ip_version
Specifies whether link stations on this port use IPv4 or IPv6 addresses. All link stations that use the port must use the same type of address. Possible values:

IPV4 Link stations on this port use IPv4 addresses.

IPV6 Link stations on this port use IPv6 addresses.

implicit_cp_cp_sess_support
Specifies whether CP-CP sessions are allowed for implicit link stations using this port. Possible values are:

YES CP-CP sessions are allowed for implicit link stations.

NO CP-CP sessions are not allowed for implicit link stations.

implicit_limited_resource
Specifies whether implicit link stations off this port are defined as limited resources. Possible values are:

NO Implicit links are not limited resources and are not automatically deactivated.

NO_SESSIONS
Implicit links are limited resources and are automatically deactivated when no active sessions are using them.

INACTIVITY
Implicit links are limited resources and are automatically

deactivated when no active sessions are using them or when no data has flowed for the time period specified by the *implicit_deact_timer* parameter.

- If no SSCP-PU session is active across the link, the node deactivates the link immediately.
- If an SSCP-PU session is active but no traffic has flowed for the specified time period, the node sends REQDISCONT(normal) to the host. The host is then responsible for deactivating all LUs and the PU, at which time the local node will deactivate the link. However, the host may not deactivate LUs with active PLU-SLU sessions; in this case, the link remains active until all these sessions are deactivated (for example by the user logging out). This behavior can be changed by using options in the *ptf* parameter of the **start_node** command.

implicit_deact_timer
Implicit limited resource link deactivation timer, in seconds.

If *implicit_limited_resource* is set to `NO_SESSIONS`, then the implicit link is automatically deactivated if no data flows on it for the time specified by this parameter and no sessions are using the link.

If *implicit_limited_resource* is set to `INACTIVITY`, an implicit link using this port is automatically deactivated if no data flows on it for the time specified by this parameter.

The minimum value is `5`; values in the range `1`–`4` will be interpreted as `5`. The value `0` (zero) indicates no timeout (the link is not deactivated, as though *implicit_limited_resource* were set to `NO`). This parameter is reserved if *implicit_limited_resource* is set to `NO`.

implicit_uplink_to_en
This parameter applies only if the local node is a Branch Network Node; it is reserved if the local node is any other type.

If the adjacent node is an end node, this parameter specifies whether implicit link stations off this port are uplink or downlink. This parameter is ignored if there are existing links to the same adjacent node, because in this case the existing links are used to determine the link type. Possible values are:

YES Implicit links to an End Node are uplinks.

NO Implicit links to an End Node are downlinks.

effect_cap **through** *user_def_parm_3*
Default TG characteristics used for implicit link stations using this port and as the default TG characteristics for defined link stations that do not have explicitly defined TG characteristics. The TG characteristics parameters are ignored if the LS is to a downstream PU.

For more information about these parameters, see “define_ip_ls” on page 67.

initially_active
Specifies whether this port is automatically started when the node is started. Possible values are:

YES The port is automatically started when the node is started.

NO The port is automatically started only if an LS that uses the port is defined as initially active; otherwise, it must be manually started.

implicit_ls_limit
Specifies the maximum number of implicit link stations which can be active on this port simultaneously, including dynamic links and links activated for Discovery. Specify a value in the range 1–65,534 or specify 0 (zero) to indicate no limit. A value of NO_IMPLICIT_LINKS indicates that no implicit links are allowed.

ip_ack_timeout **through** *short_hold_mode*
For more information about these parameters, see "define_ip_ls" on page 67. When the LS name is not initially known, the values specified on **define_ip_port** are used as defaults for processing incoming calls.

local_ip_interface
Identifier for the local network adapter card to be used for the IP link, if you have access to multiple IP networks. If you have access to only one IP network, there is no need to specify this identifier.

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

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

To determine the interface identifier:

- Run the command **iconfig** on the server where the card is installed.
- Look for the addr or addr6 field in the output to determine the identifier to be used in this parameter.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_PORT_NAME
The value specified in the *port_name* parameter was not valid.

INVALID_DLC_NAME
The specified *dlc_name* did not match any defined DLC.

INVALID_BTU_SIZE
The *max_rcv_btu_size* parameter was not set to a valid value.

INVALID_LINK_ACTIVE_LIMIT
One of the activation limit parameters, *inb_link_act_lim*, *out_link_act_lim*, or *tot_link_act_lim*, was not set to a valid value.

INVALID_MAX_IFRM_RCVD
The *max_ifrm_rcvd* parameter was not set to a valid value.

UNKNOWN_IP_HOST
The string specified for the *remote_hostname* parameter could not be resolved to a valid IP address.

INVALID_IMPLICIT_UPLINK
The *implicit_uplink_to_en* parameter was not set to a valid value.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

PORT_ACTIVE
The specified port cannot be modified because it is currently active.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

Incoming Calls

If you are configuring a port that accepts incoming calls (as defined by the *tot_link_act_lim*, *inb_link_act_lim*, and *out_link_act_lim* parameters), there is no need to define an LS to use for these calls; Communications Server for Linux will dynamically define an LS when the incoming call is received.

When an incoming call arrives at the port, Communications Server for Linux checks the address specified on the call against the addresses specified for link stations defined on the port (if any) to determine if an LS has already been defined for the call. If the address does not match, an LS is dynamically defined. To ensure that the explicit LS definition is used, be sure that the address defined for this LS matches the address that is supplied by the remote computer on the incoming call.

define_local_lu

The **define_local_lu** command defines a new local LU. The command can also be used to modify the *disable*, *description*, *sys_name*, or *timeout* parameters for an existing LU or for the default LU associated with the local node's control point, but it cannot modify any of the other parameters. When modifying an existing LU, all the other parameters that cannot be modified must be set to their currently defined values.

Supplied Parameters

Parameter name	Type	Length	Default
[define_local_lu]			
lu_name	character	8	
description	character	31	(null string)
list_name	character	14	(null string)
lu_alias	character	8	
nau_address	decimal		0
syncpt_support	constant		NO

```
lu_session_limit                decimal                 0
default_pool                    constant                NO
pu_name                         character       8       (null string)
lu_attributes                   constant                NONE
sscp_id                         decimal                 0
disable                         constant                NO
sys_name                        character       128     (null string)
timeout                         decimal                 60
```

Supplied parameters are:

lu_name
: Name of the local LU. This name is a type-A character string starting with a letter. To modify the default LU associated with the local node's control point, do not specify this parameter.

description
: A text string describing the local LU. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_local_lu** command.

list_name
: Name of the security access list used by this local LU (defined using the **define_security_access_list** command). This parameter restricts the LU so that only the users named in the specified list can use it. To specify that the LU is available for use by any user, do not specify this parameter.

lu_alias
: Alias of the local LU. This alias is a character string using any locally displayable characters.

nau_address
: Network accessible unit address of the LU. Specify `0` (zero) if the LU is an independent LU, or an address in the range `1`–`255` if the LU is a dependent LU.

syncpt_support
: Specifies whether the LU supports sync point functions. Set this parameter to `YES` only if you have a Sync Point Manager (SPM) and Conversation Protected Resource Manager (C-PRM) in addition to the standard Communications Server for Linux product. Possible values are:

  YES Sync point functions are supported.

  NO Sync point functions are not supported.

lu_session_limit
: The maximum total number of sessions (across all modes) supported by the LU.

  For a dependent LU, this parameter must be set to `1`. For an independent LU, specify `0` (zero) for no session limit, or a value in the range `1`–`65,535`.

  If you specify an explicit limit, note the following:

  - If the LU will be communicating with parallel-session remote LUs, the session limit must include sufficient sessions for CNOS negotiation; a safe minimum is 3, or an additional 2 sessions for each partner LU.
  - The LU session limit must be greater than or equal to the sum of the session limits for all modes that the LU will use.
  - If the LU will be used by full-duplex APPC conversations, each full-duplex conversation requires two sessions.

default_pool
Specifies whether the LU is in the pool of default dependent LUs. If the LU is an independent LU, do not specify this parameter. Possible values are:

YES The LU is in the pool of default LUs, and can be used by applications that do not specify an LU name.

NO The LU is not in the pool.

pu_name
Name of the PU which this LU will use. This parameter is used only by dependent LUs; do not specify it for independent LUs. The name is a type-A character string starting with a letter.

lu_attributes
Identifies additional information about the LU. Possible values are:

NONE No additional information identified.

DISABLE_PASSWORD_SUBSTITUTION
Disable password substitution support for the local LU. Password substitution means that passwords are encrypted before transmission between the local and remote LUs, rather than being sent as clear text. Communications Server for Linux normally uses password substitution if the remote system supports it.

This value is provided as a work-around for communications with some remote systems that do not implement password substitution correctly. If you use this option, you should be aware that this involves sending and receiving passwords in clear text (which may represent a security risk). Do not set it unless there are problems with the remote system's implementation of password substitution.

sscp_id Specifies the ID of the SSCP permitted to activate this LU. This ID is a 6-byte binary string. This parameter is used only by dependent LUs, and is set to all binary zeros if the LU is an independent LU or if the LU can be activated by any SSCP.

disable Specifies whether the local LU should be disabled or enabled. Possible values are:

YES Disable the local LU.

NO Enable the local LU.

sys_name
The name of the target computer for incoming Allocate requests (requests from a partner TP to start an APPC or CPI-C conversation) that arrive at this local LU.

If the target TP is a broadcast queued TP (servers are informed of its location when it starts, so that they can route incoming Allocate requests to it), or if it always runs on the same Communications Server for Linux server as the node that owns this LU, do not specify this parameter. Otherwise, set it to the name of the computer where the TP runs.

The name must be either an alias or a fully-qualified name; you cannot specify an IP address. If the name includes a **.** (period) character, Communications Server for Linux assumes that it is a fully-qualified name; otherwise it performs a DNS lookup to determine the computer name.

timeout
The timeout value for dynamic load requests. A request will time out if the invoked TP has not issued a RECEIVE_ALLOCATE (APPC), Accept_Conversation, or Accept_Incoming (CPI-C) verb within this time. Specify the timeout value in seconds, or specify -1 to indicate no timeout (dynamic load requests will wait indefinitely).

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_DISABLE
The *disable* parameter was not set to a valid value.

INVALID_LU_NAME
The *lu_name* parameter contained a character that was not valid.

INVALID_NAU_ADDRESS
The *nau_address* parameter was not in the valid range.

INVALID_SESSION_LIMIT
The *lu_session_limit* parameter was not in the valid range.

INVALID_TIMEOUT
The *timeout* parameter was not in the valid range.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

INVALID_LU_NAME
The *lu_name* parameter contained a character that was not valid.

LU_ALREADY_DEFINED
An LU with this name has already been defined. You cannot use this command to modify any parameters of an existing LU except for the Attach routing data.

PU_NOT_DEFINED
The *pu_name* parameter did not match any defined PU name.

SECURITY_LIST_NOT_DEFINED
The *security_list_name* parameter did not match any defined security access list name.

LU_ALIAS_ALREADY_USED
An LU with this alias has already been defined. You cannot use this command to modify any parameters of an existing LU except for the Attach routing data.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_ls_routing

The **define_ls_routing** command defines the location of a partner LU using a link station.

Note: You cannot use **define_ls_routing** with an Enterprise Extender (HPR/IP) link station. This is because all traffic on these link types must flow over an RTP connection, which is not fixed to a particular link station and can switch to a different path.

Supplied Parameters

Parameter name	Type	Length	Default
[define_ls_routing]			
lu_name	character	8	
fq_partner_lu	character	17	
wildcard_fqplu	constant		NO
ls_name	character	8	

Supplied parameters are:

lu_name
Name of the local LU that will communicate with the partner LU (specified by the *fq_partner_lu* parameter) over the link specified by the *ls_name* parameter. This name is an 8–byte type-A character string.

fq_partner_lu
Fully qualified name of the partner LU with which the local LU (specified by the *lu_name* parameter) will communicate over the link specified by the *ls_name* parameter. Specify 3–17 type-A characters that consists of a 1–8 character network name, followed by a period, followed by a 1–8 character partner LU name.

You can specify a partial or full wildcard partner LU name by specifying only part of the name and setting the *wildcard_fqplu* parameter to `YES`. For example:

- `APPN.NEW` matches `APPN.NEW1`, `APPN.NEWLU`, and so on
- `APPN.` matches any LU with a network name of `APPN`, regardless of its LU name
- `APPN` matches any LU with a network name beginning with `APPN`: `APPN.NEW1`, `APPNNEW.LUTWO`, and so on.

To specify a full wildcard entry, so that all partner LUs will be accessed using the same link, set *wildcard_fqplu* to `YES` and do not specify*fq_partner_lu*.

wildcard_fqplu
Wildcard partner LU flag indicating whether the *fq_partner_lu* parameter contains a full or partial wildcard. Possible values are:

YES The *fq_partner_lu* parameter contains a wildcard entry.

NO The *fq_partner_lu* parameter does not contain a wildcard entry.

ls_name
Name of the link station to use for communication between the local LU (specified by the *lu_name* parameter) and the partner LU (specified in the *fq_partner_lu* parameter). Specify 1–8 locally displayable characters.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_LU_NAME
The *lu_name* parameter contained a character that was not valid.

INVALID_PLU_NAME
The *fq_partner_lu* parameter contained a character that was not valid or the name was not fully qualified.

INVALID_WILDCARD_NAME
The *wildcard_fqplu* parameter was specified but the *fq_partner_lu* parameter was not a valid wildcard name.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

INVALID_LU_NAME
The local LU identified by the *lu_name* parameter does not exist.

Other Conditions

Appendix A, “Common Return Codes from snaadmin Commands,” on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_lu_0_to_3

The **define_lu_0_to_3** command 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.

If this command is used to modify an existing LU, only the *description*, *priority*, and *lu_model* parameters can be changed; all other parameters must be set to their existing values.

Supplied Parameters

Parameter name [define_lu_0_to_3]	Type	Length	Default
lu_name	character	8	
description	character	31	(null string)
nau_address	decimal		2
pool_name	character	8	(null string)
pu_name	character	8	
priority	constant		MEDIUM
lu_model	constant		UNKNOWN
sscp_id	decimal		0
timeout	decimal		0
term_method	constant		
disconnect_on_unbind	constant		NO

Supplied parameters are:

lu_name
: Name of the local LU to be defined. This name is a type-A character string starting with a letter.

description
: A text string describing the LU. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_lu_0_to_3** command.

nau_address
: Network accessible unit address of the LU. This address is a number in the range 1–255.

pool_name
: Name of the pool to which this LU belongs. This name is an 8-byte type-A character string. If a pool with the specified name is not already defined, Communications Server for Linux adds a new pool with this name and assigns the LU to it.

  If the LU does not belong to a pool, do not specify this parameter.

pu_name
: Name of the PU (defined using **define_*_ls**) that this LU will use. This name is a type-A character string starting with a letter.

priority
: LU priority when sending to the host. Possible values are:

  NETWORK
  : The LU has priority on the network.

  HIGH High priority is given to the LU.

  MEDIUM Medium priority is given to the LU.

  LOW Low priority is given to the LU.

lu_model

Type of the LU. Possible values are:

3270_DISPLAY_MODEL_2
LU type is a 3270 display model 2.

3270_DISPLAY_MODEL_3
LU type is a 3270 display model 3.

3270_DISPLAY_MODEL_4
LU type is a 3270 display model 4.

3270_DISPLAY_MODEL_5
LU type is a 3270 display model 5.

PRINTER
LU type is a printer.

SCS_PRINTER
LU type is an SCS printer.

RJE_WKSTN
LU type is an RJE workstation.

UNKNOWN
LU type is unknown. LU type will be determined when the session to the host is established.

If you are not using the LU for 3270 emulation, it is not necessary to specify an explicit LU type; set this parameter to `UNKNOWN`.

If the host system supports Dynamic Definition of Dependent LUs (DDDLUs), Communications Server for Linux will define the LU dynamically on the host when the communications link to the host is established. Set this parameter to `UNKNOWN` if you want the LU model type to be defined to match the type requested by the downstream TN3270 client. Depending on the value you specify, Communications Server for Linux sends one of the following strings to the host in the DDDLU NMVT, to match the values used in the standard VTAM tables:

`3270002` for `3270_DISPLAY_MODEL_2`
`3270003` for `3270_DISPLAY_MODEL_3`
`3270004` for `3270_DISPLAY_MODEL_4`
`3270005` for `3270_DISPLAY_MODEL_5`
`3270DSC` for `PRINTER`
`3270SCS` for `SCS_PRINTER`
`3270000` for `RJE_WKSTN`
`327000`*n* for `UNKNOWN` with a TN3270 client, where *n* is the model number (2–5) provided by the client
`327000@` for `UNKNOWN` with an LUA client

If the host does not support DDDLU, the LU must be included in the host configuration.

sscp_id Specifies the ID of the SSCP permitted to activate this LU. Specify a value in the range `0`–`65,535`. If this parameter is set to `0` (zero), the LU can be activated by any SSCP.

timeout

Timeout for the LU specified in seconds. If the timeout is set to a nonzero value and the user of the LU supports session inactivity timeouts, then the LU is deactivated after the PLU-SLU session is left inactive for the specified period and one of the following conditions exist:

- The session passes over a limited resource link.
- Another application requests to use the LU before the session is used again.

If the timeout is set to `0` (zero), the LU is not deactivated.

Support for session inactivity timeouts depends on the application that is using the LU (such as a 3270 emulation program). If the LU is being used by SNA gateway, session inactivity timeouts are supported only if *allow_timeout* is specified on the **define_downstream_lu** command.

term_method
This parameter specifies how Communications Server for Linux should attempt to end a PLU-SLU session to a host from this LU. Possible values are:

USE_NODE_DEFAULT
Use the node's default termination method, specified by the *send_term_self* parameter on **define_node**.

SEND_UNBIND
End the session by sending an UNBIND.

SEND_TERM_SELF
End the session by sending a TERM_SELF.

disconnect_on_unbind
This parameter applies only when this LU is being used by a TN3270 client. It specifies whether to end the session when the host sends an UNBIND instead of displaying the VTAM MSG10 or returning to a host session manager. Possible values are:

YES End the session if the host sends an UNBIND that is not type 2 (BIND forthcoming).

NO Do not end the session if the host sends an UNBIND.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
`PARAMETER_CHECK`

secondary_rc
Possible values are:

INVALID_LU_NAME
The *lu_name* parameter contained a character that was not valid.

INVALID_POOL_NAME
The *pool_name* parameter contained a character that was not valid.

INVALID_NAU_ADDRESS
The *nau_address* parameter was not in the valid range.

INVALID_PRIORITY
The *priority* parameter was not set to a valid value.

INVALID_TERM_METHOD
The *term_method* parameter was not set to a valid value.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

INVALID_PU_NAME
The value specified in the *pu_name* parameter was not valid.

PU_NOT_DEFINED
The *pu_name* parameter did not match any defined PU name.

INVALID_PU_TYPE
The PU specified by the *pu_name* parameter is not a host PU.

LU_NAME_POOL_NAME_CLASH
The LU name matches the name of an LU pool.

LU_ALREADY_DEFD
An LU with the specified name has already been defined.

LU_NAU_ADDR_ALREADY_DEFD
An LU with the specified NAU address has already been defined.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_lu_0_to_3_range

The **define_lu_0_to_3_range** command defines a range of LUs for use with 3270 emulationor LUA (LUs of type 0, 1, 2, or 3), and optionally assigns the LUs to an LU pool. This command cannot be used to modify existing LUs.

The supplied parameters include a base name for the new LUs and the range of NAU addresses. The new LU names are generated by combining the base name with the NAU addresses (or with a defined base number). For example, a base name of LUNME combined with a NAU range of 11 to 14 would define the LUs as LUNME011, LUNME012, LUNME013, and LUNME014.

Supplied Parameters

Parameter name [define_lu_0_to_3_range]	Type	Length	Default
base_name	character	6	
description	character	31	(null string)
min_nau	decimal		1
max_nau	decimal		1
pool_name	character	8	(null string)

```
pu_name                 character      8
priority                constant                MEDIUM
lu_model                constant                UNKNOWN
sscp_id                 decimal                 0
timeout                 decimal                 0
name_attributes         constant                NONE
base_number             decimal                 0
term_method             constant
disconnect_on_unbind    constant                NO
```

Supplied parameters are:

base_name
: Base name for the names of the new LUs. This name is a type-A character string starting with a letter.

  - If the *name_attributes* parameter is set to USE_HEX_IN_NAME, this name may be up to 6 characters long. Communications Server for Linux generates the LU name for each LU by appending a 2-digit hexadecimal number to this name (starting from a base number specified by the *base_number* parameter).
  - Otherwise, this name may be up to 5 characters long. Communications Server for Linux generates the LU name for each LU by appending a 3-digit decimal number to this name (taken from the NAU address or from a defined base number, as specified by the *name_attributes* parameter).

  If USE_HEX_IN_NAME is specified for the *name_attributes* parameter, the *base_name* parameter can contain 6 characters.

description
: A text string describing the LUs; the same string is used for each LU in the range. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_lu_0_to_3** command.

min_nau
: NAU address of the first LU, in the range 1–255.

max_nau
: NAU address of the last LU, in the range 1–255.

pool_name
: Name of pool to which these LUs belong. This name is an 8-byte type-A character string. If a pool with the specified name is not already defined, Communications Server for Linux adds a new pool with this name and assigns the LUs to it.

  If the LUs do not belong to a pool, do not specify this parameter.

pu_name
: Name of the PU (defined using **define_*_ls**) that these LUs will use. This name is a type-A character string starting with a letter.

priority
: LU priority when sending to the host. Possible values are:

  NETWORK
  : The LU has priority on the network.

  HIGH
  : High priority is given to the LU.

  MEDIUM
  : Medium priority is given to the LU.

LOW Low priority is given to the LU.

lu_model

Type of the LU. Possible values are:

3270_DISPLAY_MODEL_2
LU type is a 3270 display model 2.

3270_DISPLAY_MODEL_3
LU type is a 3270 display model 3.

3270_DISPLAY_MODEL_4
LU type is a 3270 display model 4.

3270_DISPLAY_MODEL_5
LU type is a 3270 display model 5.

PRINTER
LU type is a printer.

SCS_PRINTER
LU type is an SCS printer.

RJE_WKSTN
LU type is an RJE workstation.

UNKNOWN
LU type is unknown. (LU type will be determined when the session to the host is established.)

If you are not using the LUs for 3270 emulation, it is not necessary to specify an explicit LU type; set this parameter to UNKNOWN.

If the host system supports Dynamic Definition of Dependent LUs (DDDLUs) and this parameter is set to any value other than UNKNOWN, Communications Server for Linux will define the LU dynamically on the host when the communications link to the host is established. Depending on the value you specify, Communications Server for Linux sends one of the following strings to the host in the DDDLU NMVT, to match the values used in the standard VTAM tables:

3270002 for 3270_DISPLAY_MODEL_2
3270003 for 3270_DISPLAY_MODEL_3
3270004 for 3270_DISPLAY_MODEL_4
3270005 for 3270_DISPLAY_MODEL_5
3270DSC for PRINTER
3270SCS for SCS_PRINTER
3270000 for RJE_WKSTN
327000*n* for UNKNOWN with a TN3270 client, where *n* is the model number (2–5) provided by the client
3270000 for UNKNOWN with an LUA client

If the host does not support DDDLU or if this parameter is set to UNKNOWN, the LU must be included in the host configuration.

sscp_id Specifies the ID of the SSCP permitted to activate this LU. Specify a value in the range 0–65,535. If this parameter is set to 0 (zero), the LU can be activated by any SSCP.

timeout

Timeout for the LU specified in seconds. If the timeout is set to a nonzero value and the user of the LU supports session inactivity timeouts, then the LU is deactivated after the PLU-SLU session is left inactive for the specified period and one of the following conditions exist:

- The session passes over a limited resource link.
- Another application requests to use the LU before the session is used again.

If the timeout is set to `0` (zero), the LU is not deactivated.

Support for session inactivity timeouts depends on the application that is using the LU (such as a 3270 emulation program). If the LU is being used by SNA gateway, session inactivity timeouts are supported only if *allow_timeout* is specified on the **define_downstream_lu** command.

name_attributes
Specifies the name attributes of the LUs. Possible values are:

NONE LU names have numbers that correspond to the NAU numbers. The numbers are specified in decimal and the *base_name* parameter can contain only 5 characters.

USE_BASE_NUMBER
Start naming the LUs in the range from the value specified in the *base_number* parameter.

USE_HEX_IN_NAME
Add the extension to the LU name in hex rather than decimal. The *base_name* parameter can contain 6 characters if this value is specified.

base_number
If `USE_BASE_NUMBER` is specified in the *name_attributes* parameter, specify a number from which to start naming the LUs in the range. This value will be used instead of the value of the *min_nau* parameter.

term_method
This parameter specifies how Communications Server for Linux should attempt to end a PLU-SLU session to a host from this LU. Possible values are:

USE_NODE_DEFAULT
Use the node's default termination method, specified by the *send_term_self* parameter on **define_node**.

SEND_UNBIND
End the session by sending an UNBIND.

SEND_TERM_SELF
End the session by sending a TERM_SELF.

disconnect_on_unbind
This parameter applies only when an LU in this range is being used by a TN3270 client. It specifies whether to end the session when the host sends an UNBIND instead of displaying the VTAM MSG10 or returning to a host session manager. Possible values are:

YES End the session if the host sends an UNBIND that is not type 2 (BIND forthcoming).

NO Do not end the session if the host sends an UNBIND.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_LU_NAME
The *base_name* parameter contained a character that was not valid.

INVALID_POOL_NAME
The *pool_name* parameter contained a character that was not valid.

INVALID_NAU_ADDRESS
One or more of the NAU addresses were not in the valid range.

INVALID_PRIORITY
The *priority* parameter was not set to a valid value.

INVALID_TERM_METHOD
The *term_method* parameter was not set to a valid value.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

INVALID_PU_NAME
The *pu_name* parameter value was not valid.

PU_NOT_DEFINED
The *pu_name* parameter did not match any defined PU name.

INVALID_PU_TYPE
The PU specified by the *pu_name* parameter is not a host PU.

LU_NAME_POOL_NAME_CLASH
One of the LU names in the range matches the name of an LU pool.

LU_ALREADY_DEFINED
An LU has already been defined with the name of one of the LUs in the range.

LU_NAU_ADDR_ALREADY_DEFD
An LU has already been defined with the address of one of the LUs in the range.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_lu_lu_password

The **define_lu_lu_password** command provides a password for session-level security verification between a local LU and a partner LU.

Supplied Parameters

Parameter name	Type	Length	Default
[define_lu_lu_password]			
lu_name	character	8	(null string)
lu_alias	character	8	(null string)
fqplu_name	character	17	
description	character	31	(null string)
password	hex array	8	
verification_protocol	constant		EITHER

Supplied parameters are:

lu_name
: LU name of the local LU. This name is a type-A character string. To indicate that the LU is defined by its LU alias instead of its LU name, do not specify this parameter.

lu_alias
: LU alias of the local LU. This alias is a character string using any locally displayable characters. It is used only if *lu_name* is not specified.

  If *lu_name* and *lu_alias* are not specified, the command is forwarded to the LU associated with the CP (the default LU).

fqplu_name
: Fully qualified name of the partner LU. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character partner LU name.

description
: A text string describing the password. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_lu_lu_password** command.

password
: Password. The password is an 8-byte hexadecimal string, which must not be set to all blanks or all zeros. The string must match the equivalent parameter configured for the partner LU on the remote system; however, the least significant bit of each byte is not used in session-level security verification and does not need to match.

  When you type in this parameter on the command line, the value you type in is immediately replaced by the encrypted version of the password. Therefore, the value you supply for the *password* parameter is never displayed on the command line.

verification_protocol
: Requested LU-LU verification protocol to use. Possible values are:

  BASIC Use basic LU-LU verification protocols.

ENHANCED
Use enhanced LU-LU verification protocols.

EITHER Either basic or enhanced verification is accepted.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_LU_ALIAS
The *lu_alias* parameter did not match any defined LU alias.

INVALID_LU_NAME
The *lu_name* parameter did not match any defined local LU name.

INVALID_PLU_NAME
The *fqplu_name* parameter did not match any defined partner LU name.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_lu_pool

The **define_lu_pool** command is used to define an LU pool and assign LUs to it, or to assign additional LUs to an existing pool. The LUs must be defined before adding them to the pool. You can also define a pool by specifying the pool name when defining an LU. For more information, see "define_lu_0_to_3" on page 90.

Do not use this command to remove LUs from an existing LU pool. Use **delete_lu_pool** to remove LUs and change the LU pool definition.

Supplied Parameters

Parameter name	Type	Length	Default
[define_lu_pool]			
pool_name	character	8	
description	character	31	(null string)
lu_name	character	8	(null string)

(0–10 *lu_name* parameters can be specified.)

Supplied parameters are:

pool_name
: Name of the LU pool. This name is an 8-byte type-A character string. Communications Server for Linux creates a pool with this name if one is not already defined.

description
: A text string describing the pool. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_lu_pool** command.

lu_name
: Names of the LUs that are to be assigned to the pool. To define the pool without adding any LUs, do not specify any LU names.

  Each of the specified LUs must already be defined as an LU of type 0–3. Each LU name is a type-A character string starting with a letter.

  If a specified LU is currently assigned to a different pool, Communications Server for Linux removes it from that pool (because an LU cannot be in more than one pool) and assigns it to the pool specified by this command.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
: PARAMETER_CHECK

secondary_rc
: Possible values are:

  INVALID_LU_NAME
  : One or more of the supplied LU names did not match any defined LU name.

  INVALID_POOL_NAME
  : The *pool_name* parameter contained a character that was not valid.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc

LU_NAME_POOL_NAME_CLASH
The specified pool name matches the name of an LU.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

define_lu62_timeout

The **define_lu62_timeout** command defines a timeout period for unused LU 6.2 sessions. Each timeout is for a specified resource type and resource name. If a **define_*** command is issued for a resource type and name pair already defined, the command overwrites the previous definitions. New timeout periods are only used for sessions activated after the definition is changed.

If more than one relevant timeout period is defined for a session, the shortest period applies.

Supplied Parameters

Parameter name [define_lu62_timeout]	Type	Length	Default
resource_type	constant		GLOBAL_TIMEOUT
resource_name	character	17	(null string)
timeout	decimal		

Supplied parameters are:

resource_type
Specifies the type of timeout to be defined. Possible values are:

GLOBAL_TIMEOUT
Timeout applies to all LU 6.2 sessions for the local node. The *resource_name* parameter should be set to all zeros.

LOCAL_LU_TIMEOUT
Timeout applies to all LU 6.2 sessions for the local LU specified in the *resource_name* parameter.

PARTNER_LU_TIMEOUT
Timeout applies to all LU 6.2 sessions to the partner LU specified in the *resource_name* parameter.

MODE_TIMEOUT
Timeout applies to all LU 6.2 sessions on the mode specified in the *resource_name* parameter.

resource_name
Name of the resource being queried. This value can be one of the following:

- If *resource_type* is set to GLOBAL_TIMEOUT, do not specify this parameter.

- If *resource_type* is set to LOCAL_LU_TIMEOUT, specify 1–8 type-A characters as a local LU name.
- If *resource_type* is set to PARTNER_LU_TIMEOUT, specify the fully qualified name of the partner LU as follows: 3–17 type-A characters consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character partner LU name.
- If *resource_type* is set to MODE_TIMEOUT, specifiy 1–8 type-A characters as a mode name.

This parameter is ignored if *list_options* is set to FIRST_IN_LIST.

timeout
Timeout period in seconds. A value of 0 (zero) indicates that the session immediately becomes free.

Returned Parameters

If the command executes successfully, Communications Server for Linux returns the following parameter:

OK The command executed successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_RESOURCE_TYPE
The type of timeout defined was not valid.

INVALID_LU_NAME
The *resource_type* parameter specified an LU name that was not valid.

INVALID_PARTNER_LU
The *resource_type* parameter specified a partner LU name that was not valid.

INVALID_MODE_NAME
The *resource_type* parameter specified a mode name that was not valid.

State Check

No specific state errors, other than those listed in Appendix A, “Common Return Codes from snaadmin Commands,” on page 589, occur for this command.

Other Conditions

Appendix A, “Common Return Codes from snaadmin Commands,” on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_mode

The **define_mode** command defines a mode (a set of networking characteristics to be used by a group of sessions) or modifies a previously defined mode. You cannot modify the SNA-defined mode CPSVCMG or change the COS name used by the SNA-defined mode SNASVCMG.

If you use this command to modify an existing mode, the changes will apply to any new combination of local LU and partner LU that start to use the mode after you have made the change. However, any combination of LUs already using the mode will not pick up the changes until after the next locally or remotely initiated CNOS command.

This command can also be used to specify the default COS to which any unrecognized modes will be mapped. If no default COS is specified, the SNA-defined COS #CONNECT is used.

Supplied Parameters

Parameter name	Type	Length	Default
[define_mode]			
mode_name	character	8	
description	character	31	(null string)
max_ru_size_upp	decimal		1024
receive_pacing_win	decimal		4
default_ru_size	constant		YES
max_neg_sess_lim	decimal		32767
plu_mode_session_limit	decimal		2
min_conwin_src	decimal		1
cos_name	character	8	#CONNECT
compression	constant		PROHIBITED
auto_act	decimal		0
min_conloser_src	decimal		0
max_ru_size_low	decimal		0
max_receive_pacing_win	decimal		0
max_compress_level	constant		
max_decompress_level	constant		

Supplied parameters are:

mode_name
: Name of the mode. This name is an 8-byte type-A character string starting with a letter, or starting with # for one of the SNA-defined modes such as #INTER. For information about SNA-defined modes, see the *Communications Server for Linux Administration Guide*. If the name is shorter than eight characters, spaces are added to the right to complete the string.

  To specify the default COS that will be used for any unrecognized mode names, set this parameter to a pair of angle brackets <> (indicating an empty hexadecimal array). In this case, the *cos_name* parameter is taken as the default COS name; all other parameters supplied on this command are ignored.

description
: A text string describing the mode. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_mode_definition** and **query_mode** commands.

max_ru_size_upp
Upper bound for the maximum size of RUs sent and received on sessions in this mode. The value is used when the maximum RU size is negotiated during session activation.

Specify a value in the range 256–61,440. If the *default_ru_size* parameter is set to YES, this parameter is ignored and the value is not checked.

receive_pacing_win
Session pacing window for sessions using this mode; specify a value in the range 1–63. This is the fixed value for fixed pacing and the initial value for adaptive pacing. The session pacing window is the maximum number of frames that can be received from the partner LU before the local LU must send a response. Communications Server for Linux always uses adaptive pacing unless the adjacent node specifies that it is not supported.

default_ru_size
Specifies whether Communications Server for Linux uses the *max_ru_size_upp* and *max_ru_size_low* parameters to define the maximum RU size. Possible values are:

YES Communications Server for Linux ignores the *max_ru_size_upp* and *max_ru_size_low* parameters, and sets the upper bound for the maximum RU size to the largest value that can be accommodated in the link BTU size.

NO Communications Server for Linux uses the *max_ru_size_upp* and *max_ru_size_low* parameters to define the maximum RU size.

max_neg_sess_lim
Maximum number of sessions allowed on this mode between any local LU and partner LU. This value can be lowered for a particular LU-LU-mode combination when issuing **initialize_session_limit** or **change_session_limit**.

Specify a value in the range 0–32,767. A value of 0 indicates that Communications Server for Linux should not initiate implicit CNOS exchange when an application attempts to start a session using this mode; session limits must be specified explicitly using **initialize_session_limit**.

If the mode will be used by full-duplex APPC conversations, note that each full-duplex conversation requires two sessions.

plu_mode_session_limit
Default session limit for this mode. This parameter limits the number of sessions on this mode between any one local LU and partner LU pair. This value is used when CNOS (Change Number of Sessions) exchange is initiated implicitly.

Specify a value in the range 0–32,767 (which must not exceed the value in *max_neg_sess_lim*). A value of 0 indicates that Communications Server for Linux should not initiate implicit CNOS exchange when an application attempts to start a session using this mode; session limits must be specified explicitly using **initialize_session_limit**.

If you specify an explicit limit, the LU session limit for any LU that uses this mode must be greater than or equal to the sum of the session limits for all modes that the LU will use.

If the mode will be used by full-duplex APPC conversations, note that each full-duplex conversation requires two sessions.

min_conwin_src
Minimum number of contention winner sessions that a local LU using this mode can activate. This value is used when CNOS (Change Number of Sessions) exchange is initiated either by the remote system or implicitly by Communications Server for Linux. Specify a value in the range `0–32,767`. The sum of the *min_conwin_src* and *min_conloser_src* parameters must not exceed *plu_mode_session_limit*.

cos_name
Name of the class of service (COS) to request when activating sessions on this mode. This parameter is a type-A character string.

If the node supports mode-to-cos mapping (as defined by the *mode_to_cos_map_supp* parameter on the **define_node** command), the COS specified by this parameter must be either an SNA-defined COS or a COS previously defined by a **define_cos** command. Otherwise, you do not need to specify this parameter; Communications Server for Linux ignores it.

compression
Specifies whether sessions activated using this mode can use compression. Possible values are:

`PROHIBITED`
Compression is not supported for sessions using this mode.

`REQUESTED`
Compression is supported and requested for sessions using this mode. (It is not mandatory; compression will not be used if the BIND from the partner does not request it.)

auto_act
Specifies how many sessions to activate automatically for each pair of LUs that use this mode. This value is used when CNOS (Change Number of Sessions) exchange is initiated implicitly.

The actual number of sessions activated is the minimum of this value and the negotiated minimum number of contention winner sessions for the local LU.

Specify a value in the range `0–32,767`.

min_conloser_src
Minimum number of contention loser sessions that can be activated by any one local LU that uses this mode. This value is used when CNOS (Change Number of Sessions) exchange is initiated implicitly.

Specify a value in the range `0–32,767`. The sum of the *min_conwin_src* and *min_conloser_src* parameters must not exceed *plu_mode_session_limit*.

max_ru_size_low
Lower bound for the maximum size of RUs sent and received on sessions that use this mode. This parameter is ignored if the value of the *default_ru_size* parameter is `YES`.

Specify a value in the range `256–61,440` or the value `0` (zero), which means that there is no lower bound.

max_receive_pacing_win
Maximum session pacing window for sessions in this mode. For adaptive pacing, this value is used to limit the receive pacing window that the session will grant. For fixed pacing, this parameter is not used.

(Communications Server for Linux always uses adaptive pacing unless the adjacent node specifies that it does not support it.)

Specify a value in the range 0–32,767. Specify the value 0 (zero) to indicate that there is no limit for the pacing window.

max_compress_lvl
Specifies the maximum level of compression that Communications Server for Linux will attempt to negotiate for data flowing from the local node. Possible values are:

- NONE
- RLE
- LZ9
- LZ10

If compression is negotiated using a non-extended BIND, which does not specify a maximum compression level, RLE compression will be used.

max_decompress_lvl
Specifies the maximum level of decompression that Communications Server for Linux will attempt to negotiate for data flowing into the local node. Possible values are:

- NONE
- RLE
- LZ9
- LZ10

If compression is negotiated using a non-extended BIND, which does not specify a maximum compression level, RLE compression will be used.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

CPSVCMG_ALREADY_DEFD
The SNA-defined mode CPSVCMG cannot be changed.

INVALID_COS_SNASVCMG_MODE
The COS for the SNA-defined mode SNASVCMG cannot be changed.

INVALID_MAX_RU_SIZE_UPPER
The *max_ru_size_upp* parameter was not in the valid range.

INVALID_SNASVCMG_MODE_LIMIT
The SNA-defined mode SNASVCMG must have a session limit of 2 and *min_conwin_src* of 1 or a session limit of 1 and *min_conwin_src* of 0 (zero) or both a session limit and *min_conwin_src* of 0 (zero). The values used to define SNASVCMG were not valid.

MODE_SESS_LIM_EXCEEDS_NEG
The value specified for *plu_mode_session_limit* was larger than the value specified for *max_neg_sess_lim*.

INVALID_MAX_RU_SIZE_LOW
The *max_ru_size_low* parameter was not in the valid range.

RU_SIZE_LOW_UPPER_MISMATCH
The value specified for *max_ru_size_low* exceeded the value specified for *max_ru_size_upp*.

INVALID_MIN_CONLOSERS
The *min_conloser_src* parameter was not in the valid range, or was greater than *plu_mode_session_limit*.

INVALID_MIN_CONWINNERS
The *min_conwin_src* parameter was not in the valid range, or was greater than *plu_mode_session_limit*.

INVALID_MIN_CONTENTION_SUM
The sum of the *min_conloser_src* and *min_conwin_src* parameters exceeded the value of *plu_mode_session_limit*.

State Check
No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions
Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_mpc_dlc

The **define_mpc_dlc** command (available only with Communications Server for Linux on System z) defines a new multipath channel (MPC) DLC. This command can also be used to modify an existing DLC, if the DLC is not currently active..

You can define only one MPC DLC on the Communications Server for Linux node, although it can support multiple MPC ports. Do not attempt to issue more than one **define_mpc_dlc** command to define multiple DLCs.

Supplied Parameters

Parameter name	Type	Length	Default
[define_mpc_dlc]			
dlc_name	character	8	
description	character	31	(null string)
initially_active	constant		YES

Supplied parameters are:

dlc_name
Name of the DLC. This name is a character string using any locally displayable characters.

description
A text string describing the DLC. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_dlc** command.

initially_active
Specifies whether this DLC is automatically started when the node is started. Possible values are:

YES The DLC is automatically started when the node is started.

NO The DLC is automatically started only if a port or LS that uses it is defined to be initially active; otherwise, it must be manually started.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

INVALID_DLC_NAME
The supplied *dlc_name* parameter contained a character that was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_mpc_ls

The **define_mpc_ls** command (available only with Communications Server for Linux on System z) is used to define a new multipath channel (MPC) link station (LS) or to modify an existing one. Before issuing this command, you must define the port that this link station uses.

Only one MPC LS using each MPC port can be active at any time. You can issue more than one **define_mpc_ls** command to define multiple LSs using the same port, but you will not be able to activate more than one of them at a time.

You cannot use this command to modify the port used by an existing LS; the *port_name* specified on the command must match the previous definition of the LS. The LS can be modified only if it is not started.

Supplied Parameters

Parameter name	Type	Length	Default
[define_mpc_ls]			
ls_name	character	8	
port_name	character	8	
description	character	31	(null string)
adj_cp_name	character	17	(null string)
adj_cp_type	constant		LEARN_NODE
auto_act_supp	constant		NO
tg_number	decimal		0
limited_resource	constant		NO
disable_remote_act	constant		NO
link_deact_timer	decimal		30
default_nn_server	constant		NO
ls_attributes	constant		SNA
cp_cp_sess_support	constant		YES
use_default_tg_chars	constant		YES
effect_cap	decimal		78643200
connect_cost	decimal		0
byte_cost	decimal		0
security	constant		SEC_NONSECURE
prop_delay	constant		PROP_DELAY_MINIMUM
user_def_parm1	decimal		128
user_def_parm2	decimal		128
user_def_parm3	decimal		128
target_pacing_count	decimal		7
max_ifrm_rcvd	decimal		7
max_send_btu_size	decimal		4096
initially_active	constant		NO
react_timer	decimal		1
react_timer_retry	decimal		65535
restart_on_normal_deact	constant		NO
branch_link_type	constant		UPLINK (used only if this node is BrNN)
adj_brnn_cp_support	constant		ALLOWED (used only if this node is BrNN)

Supplied parameters are:

ls_name
: Name of link station to be defined.

port_name
: Name of the port associated with this link station. This must match the name of a defined port.

description
: A text string describing the LS. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_ls** command.

adj_cp_name
: Fully qualified name of the adjacent CP for this LS. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character CP name. This parameter is used in the following ways:

  - If the *adj_cp_type* parameter is set to NETWORK_NODE or END_NODE, and preassigned TG numbers are being used, set this parameter to the CP

name defined at the adjacent node; if the adjacent node sends a CP name during XID exchange, it will be checked against this value.

- If *adj_cp_type* is set to any other value, or if preassigned TG numbers are not being used, there is no need to specify this parameter; Communications Server for Linux checks the CP name only if one is specified.

adj_cp_type
Adjacent node type.

If preassigned TG numbers are not being used, this parameter is normally set to LEARN_NODE, indicating that the node type is unknown. Communications Server for Linux will determine the type during XID exchange.

If preassigned TG numbers are being used, you must specify the node type explicitly. You can also specify it as an additional security check if preassigned TG numbers are not being used. In this case, Communications Server for Linux will reject a connection attempt from the adjacent node if its node type does not match the one specified for this parameter.

Possible values are:

LEARN_NODE
The node type is unknown. Communications Server for Linux will determine the type during XID exchange.

END_NODE
The adjacent node is an End Node, or a Branch Network Node acting as an End Node from the local node's perspective.

NETWORK_NODE
The adjacent node is a Network Node, or a Branch Network Node acting as a Network Node from the local node's perspective.

auto_act_supp
Specifies whether the link can be activated automatically when required by a session. Possible values are:

YES The link can be activated automatically.

The reactivation timer parameters are ignored. If the LS fails, Communications Server for Linux does not attempt to reactivate it until a dependent LU application that was using the session attempts to restart the session; an LS used by independent LUs is not reactivated by Communications Server for Linux and must be manually restarted.

The following restrictions also apply:

- The LS must have a preassigned TG number defined (see the *tg_number* parameter), and *cp_cp_sess_support* must be set to NO.
- If either the local node or the adjacent node is an end node, the LS must also be defined as automatically activated at the adjacent node.

NO The link cannot be activated automatically.

tg_number
Preassigned TG number, used to represent the link when the link is activated. The node will not accept any other number from the adjacent node during activation of this link. If the adjacent node is using

preassigned TG numbers, the same TG number must be defined by the adjacent node on the adjacent link station.

Specify a number in the range 1–20, or 0 (zero) to indicate that the TG number is not preassigned and is negotiated when the link is activated.

If a preassigned TG number is defined, the *adj_cp_name* parameter must also be defined, and the *adj_cp_type* parameter must be set to either END_NODE or NETWORK_NODE.

limited_resource

Specifies whether this link station is a limited resource and is automatically deactivated when there are no sessions using the link. Link stations on a nonswitched port cannot be configured as limited resource. Possible values are:

NO The link is not a limited resource and is not automatically deactivated.

NO_SESSIONS
The link is a limited resource and is automatically deactivated when no active sessions are using it.

INACTIVITY
The link is a limited resource and is automatically deactivated when no active sessions are using it, or when no data has flowed on the link for the time period specified by the *link_deact_timer* parameter.

- If no SSCP-PU session is active across the link, the node deactivates the link immediately.
- If an SSCP-PU session is active but no traffic has flowed for the specified time period, the node sends REQDISCONT(normal) to the host. The host is then responsible for deactivating all LUs and the PU, at which time the local node will deactivate the link. However, the host may not deactivate LUs with active PLU-SLU sessions; in this case, the link remains active until all these sessions are deactivated (for example by the user logging out). This behavior can be changed by using options in the *ptf* parameter of the **start_node** command.

A limited resource link station can be configured for CP-CP session support by setting this parameter to NO_SESSIONS and *cp_cp_sess_support* to YES. In this case, if CP-CP sessions are brought up over the link, Communications Server for Linux does not treat the link as a limited resource (and therefore does not deactivate it).

disable_remote_act

Specifies whether the LS can be activated by the remote node. Possible values are:

YES The LS can be activated only by the local node; if the remote node attempts to activate the LS, Communications Server for Linux rejects the attempt.

NO The LS can be activated by the remote node.

link_deact_timer

Limited resource link deactivation timer, in seconds. A limited resource link is automatically deactivated if no data flows over the link for the time specified by this parameter. This parameter is not used if *limited_resource* is set to any value other than INACTIVITY.

The minimum value is 5; values in the range 1–4 will be interpreted as 5.

The value 0 (zero) indicates that no timeout is used (the link is not deactivated, as if *limited_resource* were set to NO).

default_nn_server
For an end node, this parameter specifies whether this is a link supporting CP-CP sessions to a network node that can act as the local node's network node server. When the local node has no CP-CP sessions to a network node server and needs to establish them, it checks this parameter on its defined link stations to find a suitable LS to activate. This enables you to specify which adjacent NNs are suitable to act as the NN server (for example, to avoid using NNs that are accessed by expensive or slow links).

Possible values are:

YES This link supports CP-CP sessions to a network node that can act as the local node's NN server; the local node can automatically activate this link if it needs to contact an NN server. The *cp_cp_sess_support* parameter must be set to YES.

NO This link does not support CP-CP sessions to a network node that can act as the local node's NN server; the local node cannot automatically activate this link if it needs to contact an NN server.

If the local node is not an end node, this parameter is ignored.

ls_attributes
Attributes of the remote system that Communications Server for Linux is communicating with.

Specify SNA unless you are communicating with a host of one of the other following types. Possible values are:

SNA Standard SNA host

FNA Fujitsu Network Architecture (VTAM-F) host

HNA Hitachi Network Architecture host

cp_cp_sess_support
Specifies whether CP-CP sessions are supported. This parameter is valid only if the adjacent node is an end node or network node (*adj_cp_type* is NETWORK_NODE, END_NODE, or LEARN_NODE); it is ignored otherwise. If both the local node and the adjacent node are network nodes, this parameter should be set to YES in order to use APPN functions between these nodes. You can set either *solicit_sscp_sessions* or *cp_cp_sess_support* but not both.

Possible values are:

YES CP-CP sessions are supported. The *solicit_sscp_sessions* parameter must be set to NO.

NO CP-CP sessions are not supported.

use_default_tg_chars
Specifies whether to use the default TG characteristics supplied on **define_mpc_port**. The TG characteristics apply only if the link is to an APPN node; this parameter, and the parameters *effect_cap* through *user_def_parm_3* parameters, are ignored otherwise. Possible values are:

YES Use the default TG characteristics; ignore the parameters *effect_cap* through *user_def_parm_3* parameters on this command.

NO Use *effect_cap* through *user_def_parm_3* parameters on this command.

effect_cap
A decimal value representing the line speed in bits per second.

connect_cost
Cost per connect time. Valid values are integer values in the range `0`–`255`, where `0` is the lowest cost per connect time and `255` is the highest cost per connect time.

byte_cost
Cost per byte. Valid values are integer values in the range `0`–`255`, where `0` is the lowest cost per byte and `255` is the highest cost per byte.

security
Security level of the network. Possible values are:

SEC_NONSECURE
No security.

SEC_ENCRYPTED
Data is encrypted before transmission over the channel.

SEC_PUBLIC_SWITCHED_NETWORK
Data is transmitted over a public switched network.

SEC_UNDERGROUND_CABLE
Data is transmitted over secure underground cable.

SEC_SECURE_CONDUIT
Data is transmitted over a line in a secure conduit that is not guarded.

SEC_GUARDED_CONDUIT
Data is transmitted over a line in a conduit that is protected against physical tapping.

SEC_GUARDED_RADIATION
Data is transmitted over a line that is protected against physical and radiation tapping.

prop_delay
Propagation delay (the time a signal takes to travel the length of the link). Specify one of the following values, according to the type of link:

PROP_DELAY_MINIMUM
Minimum propagation delay

PROP_DELAY_LAN
Delay is less than .5 microseconds (typical for a LAN).

PROP_DELAY_TELEPHONE
Delay is in the range .5–50 microseconds (typical for a telephone network).

PROP_DELAY_PKT_SWITCHED_NET
Delay is in the range 50–250 microseconds (typical for a packet-switched network).

PROP_DELAY_SATELLITE
Delay is greater than 250 microseconds (typical for a satellite link).

PROP_DELAY_MAXIMUM
Maximum propagation delay

user_def_parm_1 **through** *user_def_parm_3*
User-defined parameters that you can use to include TG characteristics not covered by the previously described parameters. Each of these parameters must be set to a value in the range 0–255.

target_pacing_count
Indicates the desired pacing window size. Specify a value in the range 1–32,767.

max_ifrm_rcvd
Maximum number of I-frames that can be received by the local link stations before an acknowledgment is sent. Specify a value in the range 0–127.

max_send_btu_size
Maximum BTU size that can be sent from this link station. This value is used to negotiate the maximum BTU size that a pair of link stations can use to communicate with each other. The value includes the length of the TH and RH (total 9 bytes), as well as the RU. Specify a value in the range 265–4096.

initially_active
Specifies whether this LS is automatically started when the node is started. Possible values are:

YES The LS is automatically started when the node is started.

NO The LS is not automatically started; it must be manually started.

react_timer
Reactivation timer for reactivating a failed LS. If the *react_timer_retry* parameter has a nonzero value (to specify that Communications Server for Linux should retry activating the LS if it fails), this parameter specifies the time in seconds between retries. When the LS fails, or when an attempt to reactivate it fails, Communications Server for Linux waits for the specified time before retrying the activation. If *react_timer_retry* is 0 (zero), this parameter is ignored.

react_timer_retry
Retry count for reactivating a failed LS. This parameter is used to specify whether Communications Server for Linux should attempt to reactivate the LS if it fails while in use (or if an attempt to start the LS fails).

Specify 0 (zero) to indicate that Communications Server for Linux should not attempt to reactivate the LS, or specify the number of retries to make. A value of 65,535 indicates that Communications Server for Linux should retry indefinitely until the LS is reactivated.

Communications Server for Linux waits the time specified by the *react_timer* parameter between successive retries. If the LS is not successfully reactivated by the end of the retry count, or if **stop_ls** is issued while Communications Server for Linux is retrying the activation, no further retries are made; the LS remains inactive unless **start_ls** is issued for it.

If the *auto_act_supp* parameter is set to YES, the *react_timer* and *react_timer_retry* parameters are ignored; if the link fails, Communications Server for Linux does not attempt to reactivate it until the user application that was using the session attempts to restart the session.

restart_on_normal_deact
Specifies whether Communications Server for Linux should attempt to reactivate the LS if it is deactivated normally by the remote system. Possible values are:

YES If the remote system deactivates the LS normally, Communications Server for Linux attempts to reactivate it, using the same retry timer and count values as for reactivating a failed LS (the *react_timer* and *react_timer_retry* parameters above).

NO If the remote system deactivates the LS normally, Communications Server for Linux does not attempt to reactivate it.

If the LS is a host link (specified by the *adj_cp_type* parameter), or is automatically started when the node is started (the *initially_active* parameter is set to YES), this parameter is ignored; Communications Server for Linux always attempts to reactivate the LS if it is deactivated normally by the remote system (unless *react_timer_retry* is zero).

branch_link_type
This parameter applies only if the local node is a Branch Network Node; it is not used if the local node is any other type.

If the parameter *adj_cp_type* is set to NETWORK_NODE, END_NODE, APPN_NODE, or BACK_LEVEL_LEN_NODE, this parameter defines whether the link is an uplink or a downlink. Possible values are:

UPLINK The link is an uplink.

DOWNLINK
The link is a downlink.

If *adj_cp_type* is set to NETWORK_NODE, this parameter must be set to UPLINK.

adj_brnn_cp_support
This parameter applies only if the local node is a Branch Network Node and the adjacent node is a network node (the parameter *adj_cp_type* is set to NETWORK_NODE, or it is set to APPN_NODE and the node type discovered during XID exchange is network node). It is not used if the local and remote nodes are any other type.

This parameter defines whether the adjacent node can be a Branch Network Node that is acting as a Network Node from the point of view of the local node. Possible values are:

ALLOWED
The adjacent node is allowed (but not required) to be a Branch Network Node.

REQUIRED
The adjacent node must be a Branch Network Node.

PROHIBITED
The adjacent node must not be a Branch Network Node.

If *adj_cp_type* is set to NETWORK_NODE and *auto_act_supp* is set to YES, this parameter must be set to REQUIRED or PROHIBITED.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

CANT_MODIFY_PORT_NAME
The *ls_name* parameter matched the name of an existing LS, but the *port_name* parameter did not match the existing definition. You cannot modify the port name when changing the definition of an existing LS.

DEF_LINK_INVALID_SECURITY
The *security* parameter was not set to a valid value.

INVALID_AUTO_ACT_SUPP
The *auto_act_supp* parameter was not set to a valid value, or was set to YES when *cp_cp_sess_support* was also set to YES.

INVALID_CP_NAME
The *adj_cp_name* parameter contained a character that was not valid, was not in the correct format, or was not specified when required.

INVALID_LIMITED_RESOURCE
The *limited_resource* parameter was not set to a valid value.

INVALID_LINK_NAME
The *ls_name* parameter contained a character that was not valid.

INVALID_NODE_TYPE
The *adj_cp_type* parameter was not set to a valid value.

INVALID_PORT_NAME
The *port_name* parameter did not match the name of any defined port.

INVALID_PU_NAME
The *pu_name* parameter did not match the name of any defined PU, or was set to a new value on an already-defined LS.

INVALID_SOLICIT_SSCP_SESS
The *solicit_sscp_sess* parameter was not set to a valid value.

INVALID_TARGET_PACING_CNT
The *target_pacing_count* parameter was not set to a valid value.

INVALID_TG_NUMBER
The *tg_number* parameter value was not in the valid range.

MISSING_CP_NAME
A TG number was defined, but no CP name was supplied.

MISSING_CP_TYPE
A TG number was defined, but no CP type was supplied.

MISSING_TG_NUMBER
The link was defined as automatically activated, but no TG number was supplied.

INVALID_BRANCH_LINK_TYPE
The *branch_link_type* parameter was not set to a valid value.

INVALID_BRNN_SUPPORT
The *adj_brnn_cp_support* parameter was not set to a valid value.

BRNN_SUPPORT_MISSING
The *adj_brnn_cp_support* parameter was set to ALLOWED; this value is not valid because the adjacent node is a Network Node and *auto_act_supp* is set to YES.

INVALID_UPLINK
The *branch_link_type* parameter was set to UPLINK, but the definition of an existing LS between the local and adjacent nodes specifies that it is a downlink. The branch link type must be the same for all LSs between the same two nodes.

INVALID_DOWNLINK
The *branch_link_type* parameter was set to DOWNLINK, but the definition of an existing LS between the local and adjacent nodes specifies that it is an uplink. The branch link type must be the same for all LSs between the same two nodes.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

DUPLICATE_CP_NAME
The CP name specified in the *adj_cp_name* parameter has already been defined.

INVALID_LINK_NAME
The link station value specified in the *ls_name* parameter was not valid.

INVALID_NUM_LS_SPECIFIED
The number of link stations specified was not valid.

LOCAL_CP_NAME
Adjacent CP name matches local CP name.

LS_ACTIVE
The link station specified in the *ls_name* parameter is currently active.

PU_ALREADY_DEFINED
The PU specified in the *pu_name* parameter has already been defined.

DUPLICATE_TG_NUMBER
The TG number specified in the *tg_number* parameter has already been defined.

TG_NUMBER_IN_USE
TG number in use.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_mpc_port

The **define_mpc_port** command (available only with Communications Server for Linux on System z) is used to define a new multipath channel (MPC) port or modify an existing one. Before issuing this command, you must define the DLC that this port uses.

You can modify an existing port only if it is not started. You cannot change the DLC used by an existing port; the *dlc_name* specified when modifying an existing port must match the DLC name that was specified on the initial definition of the port.

Supplied Parameters

Parameter name	Type	Length	Default
[define_mpc_port]			
port_name	character	8	
dlc_name	character	8	
description	character	31	(null string)
port_number	decimal		0
max_rcv_btu_size	decimal		4096
inb_link_act_lim	decimal		0
out_link_act_lim	decimal		0
act_xid_exchange_limit	decimal		9
nonact_xid_exchange_limit	decimal		5
max_ifrm_rcvd	decimal		7
target_pacing_count	decimal		7
max_send_btu_size	decimal		4096
implicit_cp_cp_sess_support	constant		YES
implicit_limited_resource	constant		NO
implicit_deact_timer	decimal		30
implicit_ls_limit	decimal		0
implicit_uplink_to_en	constant		NO
effect_cap	decimal		78643200
connect_cost	decimal		0
byte_cost	decimal		0
security	constant		SEC_NONSECURE
prop_delay	constant		PROP_DELAY_MINIMUM
user_def_parm_1	decimal		128
user_def_parm_2	decimal		128
user_def_parm_3	decimal		128
initially_active	constant		YES

Supplied parameters are:

port_name
Name of the port to be defined. This name is a character string using any locally displayable characters.

dlc_name
Name of the associated DLC. This name is a character string using any locally displayable characters. The specified DLC must have already been defined.

description
A text string describing the port. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_port** command.

port_number
The number of the port. This is the number corresponding to the MultiPath Channel device: for example, port 0 is **/dev/mpc0** and port 1 is **/dev/mpc1**.

max_rcv_btu_size
Maximum BTU size that can be received. This value includes the length of the TH and RH (total 9 bytes) as well as the RU. Specify a value in the range 265–4096.

inb_link_act_lim
Inbound link activation limit (the number of links reserved for inbound activation). This must be set to 0 or 1.

out_link_act_lim
Outbound link activation limit (the number of links reserved for outbound activation). This must be set to 0 or 1.

act_xid_exchange_limit
Activation XID exchange limit. Specify a value in the range 0–65,535.

nonact_xid_exchange_limit
Nonactivation XID exchange limit. Specify a value in the range 0–65,535.

max_ifrm_rcvd
Maximum number of I-frames that can be received by the local link stations before an acknowledgment is sent. Specify a value in the range 1–127.

target_pacing_count
Indicates the desired pacing window size. Specify a value in the range 1–32,767.

max_send_btu_size
Maximum BTU size that can be sent from this port. This value is used to negotiate the maximum BTU size that a pair of link stations can use to communicate with each other. The value includes the length of the TH and RH (total 9 bytes) as well as the RU. Specify a value in the range 265–4096.

implicit_cp_cp_sess_support
Specifies whether CP-CP sessions are allowed for implicit link stations using this port. Possible values are:

YES CP-CP sessions are allowed for implicit link stations.

NO CP-CP sessions are not allowed for implicit link stations.

implicit_limited_resource
Specifies whether implicit link stations off this port should be defined as limited resources. Possible values are:

NO Implicit links are not limited resources, and will not be deactivated automatically.

NO_SESSIONS
Implicit links are limited resources, and will be deactivated automatically when no active sessions are using them.

INACTIVITY
Implicit links are limited resources, and will be deactivated automatically when no active sessions are using them or when no data has been transmitted for the time period specified by the *implicit_deact_timer* parameter.

implicit_deact_timer
Implicit limited resource link deactivation timer, in seconds.

If *implicit_limited_resource* is set to INACTIVITY, an implicit link using this port is automatically deactivated if no data flows on it for the time specified by this parameter.

The minimum value is 5; values in the range 1–4 will be interpreted as 5. The value 0 (zero) indicates no timeout (the link is not deactivated, as though *implicit_limited_resource* were set to NO).

implicit_ls_limit
Specifies the maximum number of implicit link stations which can be active on this port simultaneously, including dynamic links and links activated for Discovery. Specify 1, or specify 0 (zero) to indicate no limit. A value of NO_IMPLICIT_LINKS indicates that no implicit links are allowed.

implicit_uplink_to_en
This parameter applies only if the local node is a Branch Network Node; it is reserved if the local node is any other type.

If the adjacent node is an end node, this parameter specifies whether implicit link stations off this port are uplink or downlink. This parameter is ignored if there are existing links to the same adjacent node, because in this case the existing links are used to determine the link type. Possible values are:

YES Implicit links to an End Node are uplinks.

NO Implicit links to an End Node are downlinks.

effect_cap
A decimal value representing the line speed in bits per second.

connect_cost
Cost per connect time. Valid values are integer values in the range 0–255, where 0 is the lowest cost per connect time and 255 is the highest cost per connect time.

byte_cost
Cost per byte. Valid values are integer values in the range 0–255, where 0 is the lowest cost per byte and 255 is the highest cost per byte.

security
Security level of the network. Possible values are:

SEC_NONSECURE
No security.

SEC_PUBLIC_SWITCHED_NETWORK
Data is transmitted over a public switched network.

SEC_UNDERGROUND_CABLE
Data is transmitted over secure underground cable.

SEC_SECURE_CONDUIT
Data is transmitted over a line in a secure conduit that is not guarded.

SEC_GUARDED_CONDUIT
Data is transmitted over a line in a conduit that is protected against physical tapping.

SEC_ENCRYPTED
Data is encrypted before transmission over the channel.

SEC_GUARDED_RADIATION
Data is transmitted over a line that is protected against physical and radiation tapping.

prop_delay
Propagation delay (the time that a signal takes to travel the length of the link). Specify one of the following values, according to the type of link:

PROP_DELAY_MINIMUM
Minimum propagation delay.

PROP_DELAY_LAN
Delay is less than .5 microseconds (typical for a LAN).

PROP_DELAY_TELEPHONE
Delay is in the range .5–50 microseconds (typical for a telephone network).

PROP_DELAY_PKT_SWITCHED_NET
Delay is in the range 50–250 microseconds (typical for a packet-switched network).

PROP_DELAY_SATELLITE
Delay is greater than 250 microseconds (typical for a satellite link).

PROP_DELAY_MAXIMUM
Maximum propagation delay.

user_def_parm_1 **through** *user_def_parm_3*
User-defined parameters that you can use to include TG characteristics not covered by the previously described parameters. Each of these parameters must be set to a value in the range 0–255.

initially_active
Specifies whether this port is automatically started when the node is started. Possible values are:

YES The port is automatically started when the node is started.

NO The port is automatically started only if an LS that uses it is defined to be initially active; otherwise, it must be manually started.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_PORT_NAME
The value specified in the *port_name* parameter was not valid.

INVALID_DLC_NAME
The specified *dlc_name* did not match any defined DLC.

INVALID_BTU_SIZE
The *max_rcv_btu_size* parameter was not set to a valid value.

INVALID_LINK_ACTIVE_LIMIT
One of the activation limit parameters was not set to a valid value.

INVALID_MAX_IFRM_RCVD
The *max_ifrm_rcvd* parameter was not set to a valid value.

INVALID_IMPLICIT_UPLINK
The *implicit_uplink_to_en* parameter was not set to a valid value.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc

PORT_ACTIVE
The port specified by the *port_name* parameter cannot be modified because it is currently active.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_node

The **define_node** command defines a new node, or modifies an existing node. It must be issued to a server where the node is not running; it cannot be issued to a running node.

When using the command-line administration program to configure a node for the first time (creating the node's configuration file) **define_node** must be the first command issued.

There is no equivalent command to delete a node. If you want to remove the entire configuration of the node and create a completely new configuration, stop the node, and then delete or rename the node's configuration file. Next, issue a new **define_node** command to the inactive node to create a new node configuration file.

Supplied Parameters

Parameter name	Type	Length	Default
[define_node]			
node_name	character	128	(null string)
description	character	31	(null string)
node_type	constant		END_NODE
fqcp_name	character	17	
cp_alias	character	8	
mode_to_cos_map_supp	constant		YES
mds_supported	constant		YES
node_id	hex array	4	0x07100000
max_locates	decimal		1500
dir_cache_size	decimal		255
max_dir_entries	decimal		0
locate_timeout	decimal		0
reg_with_nn	constant		YES
reg_with_cds	constant		YES
mds_send_alert_q_size	decimal		100
cos_cache_size	decimal		24
tree_cache_size	decimal		40
tree_cache_use_limit	decimal		40
max_tdm_nodes	decimal		0
max_tdm_tgs	decimal		0
max_isr_sessions	decimal		1000
isr_sessions_upper_threshold	decimal		900
isr_sessions_lower_threshold	decimal		800
isr_max_ru_size	decimal		16384
isr_rcv_pac_window	decimal		8
store_endpt_rscvs	constant		NO
store_isr_rscvs	constant		NO
store_dlur_rscvs	constant		NO
cos_table_version	constant		VERSION_1_COS_TABLES
send_term_self	constant		NO
disable_branch_awareness	constant		NO
cplu_syncpt_support	constant		NO
cplu_attributes	constant		NONE
dlur_support	constant		YES
pu_conc_support	constant		YES
nn_rar	decimal		128
max_ls_exception_events	decimal		0
max_compress_level	constant		LZ10
ptf_flags	constant		NONE
clear_initial_topology	constant		NO

Supplied parameters are:

node_name
: Name of the Communications Server for Linux node to be defined. The name must match the computer name of the server where the node runs.

  In a command issued to the **snaadmin** program, this parameter is optional; if you specify it, it must match the node name to which the command is issued (specified using the **-n** command-line option).

  If the computer name includes a **.** (period) character, Communications Server for Linux assumes that it is a fully-qualified name; otherwise it performs a DNS lookup to determine the computer name.

description
: A text string describing the node. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_node** command.

node_type
: Type of the node. Possible values are:

LEN_NODE
Low entry networking (LEN) node

END_NODE
APPN end node

NETWORK_NODE
APPN network node

BRANCH_NETWORK_NODE
APPN branch network node

fqcp_name
Fully qualified CP name of the node. The name is a type-A character string, consisting of 1–8 character network name, a period character, and a 1–8 character control point (CP) name.

cp_alias
Locally used LU alias for the control point (CP) LU. This alias can be used by APPC applications to access the CP LU. This alias is a string of 1–8 characters.

mode_to_cos_map_supp
Specifies whether the node provides mode-to-COS mapping. This parameter is ignored for a network node; mode-to-COS mapping is always supported. For a LEN node, mode-to-COS mapping is not supported. Possible values are:

YES The node provides mode-to-COS mapping. A mode defined for this node must include its associated COS name, which specifies either an SNA-defined COS or a COS defined using **define_cos**.

NO The node does not provide mode-to-COS mapping. The network node server for the end node performs mode-to-COS mapping.

mds_supported
Specifies whether Management Services (MS) supports Multiple Domain Support (MDS) and Management Services Capabilities. Possible values are:

YES MDS is supported.

NO MDS is not supported.

node_id
Node identifier used in XID exchange. This ID is a 4-byte hexadecimal string, consisting of a block number (three hexadecimal digits) and a node number (five hexadecimal digits).

max_locates
Maximum number of locate requests (requests for which a response has not yet been received) that the node can process simultaneously. When the number of outstanding locate requests reaches this limit, any further locate requests are rejected. Specify a value in the range 8–65,535.

dir_cache_size
Network node only: Size of the directory cache. The minimum size is 3. You can use the information returned on **query_directory_stats** to help determine the appropriate size.

max_dir_entries
Maximum number of directory entries. Specify a value in the range 8–65,535, or 0 to indicate no limit.

locate_timeout
Specifies the time in seconds before a network search will time out. The value 0 (zero) indicates no time out.

reg_with_nn
End node only: Specifies whether to register the node's resources with the network node server when the node is started. Possible values are:

YES Register resources with the network node server. The end node's network node server will only forward directed locates to the end node.

NO Do not register resources with the network node server. The network node server will forward all broadcast searches to the end node.

reg_with_cds
End node: Specifies whether the network node server is allowed to register end node resources with a central directory server (CDS). This parameter is ignored if *reg_with_nn* is set to NO.

Network node: Specifies whether local or domain resources can be optionally registered with central directory server (CDS).

Possible values are:

YES Register resources with the CDS.

NO Do not register resources with the CDS.

mds_send_alert_q_size
Size of the MDS send alert queue. If the number of queued alerts reaches this limit, Communications Server for Linux deletes the oldest alert on the queue. The minimum number of queued alerts is 2.

cos_cache_size
Size of the COS Database weights cache. This value should be set to the maximum number of COS definitions required. Specify a value in the range 8–65,535.

tree_cache_size
Network node: Size of the Topology Database routing tree cache. The minimum is 8 entries. For an end node or LEN node, this parameter is reserved.

tree_cache_use_limit
Network node: Maximum number of uses of a cached tree. When this number is exceeded, the tree is discarded and recomputed. This enables the node to balance sessions among equal weight routes. A low value provides better load balancing at the expense of increased activation latency. The minimum number of uses is 1. For an end node or LEN node, this parameter is reserved.

max_tdm_nodes
Network node: Maximum number of nodes that can be stored in Topology Database. A value of 0 (zero) indicates an unlimited number of nodes. For an end node or LEN node, this parameter is reserved.

max_tdm_tgs
Network node: Maximum number of TGs that can be stored in Topology Database. A value of 0 (zero) indicates an unlimited number of nodes. For an end node or LEN node, this parameter is reserved.

max_isr_sessions
Network node: Maximum number of ISR sessions the node can participate in simultaneously. This parameter is reserved for an end node or LEN node.

isr_sessions_upper_threshold **and** *isr_sessions_lower_threshold*
Network node: These thresholds control the node's congestion status, which is reported to other nodes in the network for use in route calculations. The node state changes from uncongested to congested if the number of ISR sessions exceeds the upper threshold. The node state changes back to uncongested when the number of ISR sessions dips below the lower threshold. The lower threshold must be less than the upper threshold, and the upper threshold must be lower than *max_isr_sessions*. For an end node or LEN node, these parameters are reserved.

isr_max_ru_size
Network node: Maximum RU size supported for intermediate sessions. If the supplied value is not a valid RU size (as described in *Systems Network Architecture: Formats*), Communications Server for Linux will round the value up to the next valid value. For an end node or LEN node, this parameter is reserved.

isr_rcv_pac_window
Network node: Suggested receive pacing window size for intermediate sessions, in the range 1–63. This value is only used on the secondary hop of intermediate sessions if the adjacent node does not support adaptive pacing. For an end node or LEN node, this parameter is reserved.

store_endpt_rscvs
Specifies whether RSCVs for endpoint sessions are stored for diagnostic purposes. If this parameter is set to YES, an RSCV is returned on the **query_session** command. (Setting this parameter to YES means an RSCV is stored for each endpoint session. This extra storage can be up to 256 bytes per session.) Possible values are:

YES RSCVs are stored for diagnostic purposes.

NO RSCVs are not stored for diagnostic purposes.

store_isr_rscvs
Network node: Specifies whether RSCVs for ISR sessions are stored for diagnostic purposes. If this parameter is set to YES, an RSCV is returned on the **query_isr_session** command. (Setting this parameter to YES means an RSCV is stored for each Intermediate Session Routing (ISR) session. This extra storage can be up to 256 bytes per session.) Possible values are as follows:

YES RSCVs are stored for diagnostic purposes.

NO RSCVs are not stored for diagnostic purposes.

store_dlur_rscvs
Specifies whether RSCVs for each PLU-SLU session using DLUR are stored for diagnostic purposes. If this parameter is set to YES, an RSCV is returned on the **query_dlur_lu** command. (Setting this value to YES means an RSCV is stored for each PLU-SLU session using DLUR. This extra storage can be up to 256 bytes per session.) Possible values are:

YES RSCVs are stored for diagnostic purposes.

NO RSCVs are not stored for diagnostic purposes.

cos_table_version
Specifies the version of the COS tables used by the node. Specify one of the following values:

VERSION_0_COS_TABLES
Use the COS tables originally defined in the APPN Architecture Reference.

VERSION_1_COS_TABLES
Use the COS tables originally defined for HPR over ATM.

send_term_self
Specifies the default method for ending a PLU-SLU session to a host. The value you specify is used for all type 0–3 LUs on the node, unless you override it by specifying a different value in the LU definition. Specify one of the following values:

YES Send a TERM_SELF on receipt of a CLOSE_PLU_SLU_SEC_RQ.

NO Send an UNBIND on receipt of a CLOSE_PLU_SLU_SEC_RQ.

disable_branch_awareness
This parameter applies only if *node_type* is NETWORK_NODE; it is reserved for other node types.

Specify whether the local node supports branch awareness, APPN Option Set 1120, using one of the following values:

YES The local node does not support branch awareness. TGs between this node and served Branch Network Nodes do not appear in the network topology, and the local node does not report itself as being branch aware.

NO The local node supports branch awareness.

cplu_syncpt_support
Specifies whether the node's Control Point LU supports Syncpoint functions. This parameter is equivalent to the *syncpt_support* parameter on **define_local_lu**, but applies only to the node's Control Point LU (which does not have an explicit LU definition).

Set this parameter to YES only if you have a Sync Point Manager (SPM) and Conversation Protected Resource Manager (C-PRM) in addition to the standard Communications Server for Linux product. Possible values are:

YES Syncpoint is supported.

NO Syncpoint is not supported.

cplu_attributes
Identifies additional information about the node's Control Point LU. This parameter is equivalent to the *lu_attributes* parameter on **define_local_lu**, but applies only to the node's Control Point LU (which does not have an explicit LU definition).

Possible values are:

NONE No additional information identified.

DISABLE_PWSUB
Disable password substitution support for the control point LU. Password substitution means that passwords are encrypted before transmission between the local and remote LUs, rather than being

sent as clear text. Communications Server for Linux normally uses password substitution if the remote system supports it.

This value is provided as a work-around for communications with some remote systems that do not implement password substitution correctly. If you use this option, you should be aware that this involves sending and receiving passwords in clear text (which may represent a security risk). Do not set it unless there are problems with the remote system's implementation of password substitution.

dlur_support
Specifies whether DLUR is supported. For a LEN node, this parameter is reserved. Possible values are:

YES DLUR is supported.

LIMITED_MULTI_SUBNET
End Node: DLUR is supported, but will not be used to connect to a DLUS in another subnet. If multi-subnet operation is not required, you should use this value instead of YES, to reduce network traffic and congestion at the network node.

This value is not supported for a Network Node.

NO DLUR is not supported.

pu_conc_support
Specifies whether SNA gateway is supported. Possible values are:

YES SNA gateway is supported.

NO SNA gateway is not supported.

nn_rar The network node's route additional resistance. This value is used in APPN route calculations to determine if the node is useful as an intermediate hop. A high value indicates that this node is not useful as an intermediate hop. Values must be in the range `0`–`255`.

max_ls_exception_events
The maximum number of LS exception events to be recorded by the node.

max_compress_level
The maximum compression supported by the node for LU session data. This parameter must be set to `LZ10` (the default); do not attempt to set any value other than the default.

ptf_flags
Options for configuring and controlling program temporary fix (ptf) operation. Set this parameter to `NONE` if none of the options described are required, or to one or more of the values described as follows. If two or more values are required, combine them with a + character.

Available options are:

NONE None of the options described following are required.

OVERRIDE_ERP
Communications Server for Linux normally processes an ACTPU(ERP) as an ERP; this resets the PU-SSCP session but does not implicitly deactivate the subservient LU-SSCP and PLU-SLU sessions. SNA implementations may legally process ACTPU(ERP) as if it were ACTPU(cold), implicitly deactivating the subservient

LU-SSCP and PLU-SLU sessions. To override the default processing and process all ACTPU requests as ACTPU(cold), use the value OVERRIDE_ERP.

SUPPRESS_BIS
Communications Server for Linux normally uses the BIS protocol prior to deactivating a limited resource LU 6.2 session. To suppress use of the BIS protocol so that limited resource LU 6.2 sessions are deactivated immediately using UNBIND(cleanup), use the value SUPPRESS_BIS.

OVERRIDE_REQDISCONT
Communications Server for Linux normally uses REQDISCONT to deactivate limited resource host links that are no longer required by session traffic.

If OVERRIDE_REQDISCONT is specified, it must be combined with one or both of the values IMMEDIATE_DISCONTACT and IMMEDIATE_RECONTACTto alter the type of the REQDISCONT message.

IMMEDIATE_DISCONTACT
Use type "immediate" on REQDISCONT; if this value is not specified, Communications Server for Linux uses type "normal."

IMMEDIATE_RECONTACT
Use type "immediate recontact" on REQDISCONT; if this value is not specified, Communications Server for Linux uses type "no immediate recontact."

SUPPRESS_REQDISCONT
Limited resource host links are deactivated without sending REQDISCONT.

ALLOW_BB_RQE
Communications Server for Linux normally rejects, with sense code 2003, any begin bracket (BB) exception (RQE) request from a host unless the host follows the SNA protocol that the request must also indicate change direction (CD) . Setting this flag enables Communications Server for Linux to continue sessions with hosts that do not follow this protocol.

EXTERNAL_APINGD
Communications Server for Linux normally includes a partner program for the APING connectivity tester. Setting this value disables the APING Daemon within the node. Requests by an APING program arriving at the node will not be processed automatically.

DLUR_UNBIND_ON_DACTLU
Communications Server for Linux does not normally end the PLU-SLU session when it receives a DACTLU from the host for a session using DLUR. If this value is set, Communications Server for Linux will end the PLU-SLU session when it receives a DACTLU from the host for a session using DLUR.

SUPPRESS_PU_NAME_ON_REQACTPU
Communications Server for Linux identifies the PU name in the REQACTPU message when activating DLUR PUs. Set this flag to suppress sending this identification.

LUA_PASSTHRU_BB_RACE
If an RUI application is using bracket protocols, and the host sends a BB (Begin Bracket) after the RUI application has already sent one, Communications Server for Linux normally rejects this with sense data of 0813 and does not pass it to the application. If this value is set, Communications Server for Linux passes the BB through to the RUI application. The application should send a negative response with sense data of either 0813 or 0814.

CN_OVERRIDE_LIM_RES
A link in Communications Server for Linux that uses a connection network is normally a limited resource. Set this flag to override this and instead determine whether it is a limited resource using the *implicit_limited_resource* parameter in the port associated with each connection network link.

NO_TCPIP_VECTOR
Communications Server for Linux normally includes the TCP/IP Information Control Vector (0x64) in a NOTIFY request to the host for a TN3270 or LUA session. This vector contains information that can be displayed on the host console or used by the host (for example in billing): the TCP/IP address and port number used by the client, and the IP name corresponding to the client address.

If the client address is an IPv6 address but the host is running a back-level version of VTAM that cannot interpret IPv6 addresses, the client address may be displayed incorrectly on the host console.

In some cases, for example if the host is running an older version of VTAM that does not support this vector, you may need to override this behavior so that the vector is not sent. Set this flag to suppress sending the vector to the host.

NO_TCPIP_NAME
Communications Server for Linux TN Server normally performs a Domain Name Server (DNS) lookup to determine the client IP name for inclusion in the TCP/IP Information Control Vector (0x64) as described above. You may want to avoid this DNS lookup if the DNS environment is slow, or if you know that the clients are not included in the DNS data (for example if they are DHCP clients without DDNS). To do this, set this flag to suppress the DNS lookup; Communications Server for Linux TN Server will send the CV64 control vector with the client IP address but no IP name.

This value applies only to TN3270; no DNS lookup is required for LUA clients.

clear_initial_topology
Specifies whether starting the node clears any topology data that was stored when it was last active. Possible values are:

YES Clear the stored topology data.

NO Keep any topology data that was stored when the node was last active, so that it can be re-used.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_ISR_THRESHOLDS
The ISR threshold parameters were not valid (lower threshold not less than upper, or upper threshold not less than *max_isr_sessions*).

INVALID_NODE_NAME
The *node_name* parameter contained a character that was not valid.

INVALID_CP_NAME
The *cp_alias* or *fqcp_name* parameter contained a character that was not valid.

INVALID_NODE_TYPE
The *node_type* parameter was not set to a valid value.

PU_CONC_NOT_SUPPORTED
This version of Communications Server for Linux does not support the SNA gateway feature.

DLUR_NOT_SUPPORTED
This version of Communications Server for Linux does not support the DLUR feature.

INVALID_REG_WITH_NN
The *reg_with_nn* parameter was not set to a valid value.

INVALID_COS_TABLE_VERSION
The *cos_table_version* parameter was not set to a valid value.

INVALID_SEND_TERM_SELF
The *send_term_self* parameter was not set to a valid value.

INVALID_DISABLE_BRANCH_AWRN
The *disable_branch_awareness* parameter was not set to a valid value.

INVALID_DLUR_SUPPORT
The *dlur_support* parameter was not set to a valid value.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

NODE_ALREADY_STARTED
The target node is active, so you cannot use this command to modify its configuration. The **define_node** command can be issued only to an inactive node.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists additional combinations of primary and secondary return codes that are common to all commands.

define_partner_lu

The **define_partner_lu** command defines the parameters of a partner LU for LU-LU sessions between a local LU and the partner LU, or modifies an existing partner LU. You cannot change the partner LU alias of an existing partner LU.

There is normally no requirement to define partner LUs because Communications Server for Linux will set up an implicit definition when the session to the partner LU is established; you usually only need to define the LU if you need to enforce non-default values for logical record size, conversation security support, or parallel session support. You may also have an APPC application that uses a partner LU alias when allocating a session, therefore you need to define a partner LU in order to map the alias to a fully-qualified partner LU name.

If the local node or the remote node where the partner LU is located is a LEN node, you need to define a directory entry for the partner LU to allow Communications Server for Linux to access it. To do this, use **define_adjacent_len_node**. If both the local and remote nodes are network nodes, or if one is a network node and the other is an end node, the directory entry is not required, because Communications Server for Linux can locate the LU dynamically.

Supplied Parameters

Parameter name [define_partner_lu]	Type	Length	Default
fqplu_name	character	17	
plu_alias	character	8	
description	character	31	(null string)
plu_un_name	character	8	(take from second part of fqplu_name)
max_mc_ll_send_size	decimal		0
conv_security_ver	constant		NO
parallel_sess_supp	constant		YES

Supplied parameters are:

fqplu_name
Fully qualified name of the partner LU. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character partner LU name.

plu_alias
LU alias of the partner LU. This alias is a character string using any locally displayable characters.

If the *fqplu_name* parameter matches the fully qualified name of an existing partner LU, this parameter must match the partner LU alias in the existing definition. You cannot change the partner LU alias for an existing partner LU, or set up more than one LU alias for the same fully qualified name.

Also, the partner LU alias cannot match the alias of other partner LUs or local LUs or an error code is returned.

description
A text string describing the partner LU. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_partner_lu** and **query_partner_lu_definition** commands.

plu_un_name
Uninterpreted name of the partner LU (the name of the LU as defined to the remote SSCP). The name is a type-A character string.

To use the default uninterpreted name (the same as the network name taken from the *fqplu_name* parameter), do not specify this parameter. This parameter is only relevant if the partner LU is on a host and dependent LU 6.2 is used to access it.

max_mc_ll_send_size
The maximum size of logical records that can be sent and received by mapped conversation services at the partner LU. Specify a number in the range 1–32,767, or 0 (zero) to specify no limit (in this case the maximum is 32,767).

conv_security_ver
Specifies whether the partner LU is authorized to validate user IDs on behalf of local LUs (whether the partner LU can set the already verified indicator in an Attach request). Possible values are:

YES The partner LU is authorized to validate user IDs.

NO The partner LU is not authorized to validate user IDs.

parallel_sess_supp
Specifies whether the partner LU supports parallel sessions. Possible values are:

YES The partner LU supports parallel sessions.

NO The partner LU does not support parallel sessions.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

DEF_PLU_INVALID_FQ_NAME
The *fqplu_name* parameter contained a character that was not valid.

INVALID_UNINT_PLU_NAME
The *plu_un_name* parameter contained a character that was not valid.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

PLU_ALIAS_CANT_BE_CHANGED
The *plu_alias* parameter of an existing partner LU cannot be changed.

PLU_ALIAS_ALREADY_USED
The *plu_alias* parameter is already used for an existing partner LU or local LU with a different LU name.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_qllc_dlc

The **define_qllc_dlc** command defines a new QLLC DLC. The command can also be used to modify an existing DLC, if the DLC is not currently active. However, you cannot change the negotiable link support for an existing DLC; this parameter can be specified only when creating a new DLC.

Supplied Parameters

Parameter name	Type	Length	Default
[define_qllc_dlc]			
dlc_name	character	8	
description	character	31	(null string)
adapter_number	decimal		0
initially_active	constant		YES

Supplied parameters are:

dlc_name
Name of the DLC. This name is a character string using any locally displayable characters.

description
A text string describing the DLC. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_dlc** command.

adapter_number
Adapter number used by the DLC. If the server contains more than one QLLC adapter card, specify 0 (zero) for the first card, 1 for the second card, and so on. Otherwise, set this parameter to 0 (zero).

initially_active
Specifies whether this DLC is automatically started when the node is started. Possible values are:

YES The DLC is automatically started when the node is started.

NO The DLC is automatically started only if a port or LS that uses it is defined as initially active; otherwise, it must be manually started.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

INVALID_DLC_NAME
The supplied *dlc_name* parameter contained a character that was not valid.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

DLC_ACTIVE
The DLC cannot be modified because it is currently active.

NVALID_DLC_TYPE
You cannot change the negotiable link support for an existing DLC. This parameter can be specified only when creating a new DLC.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_qllc_ls

The **define_qllc_ls** command is used to define a new QLLC link station (LS) or modify an existing one. Before issuing this command, you must define the port that this LS uses.

You cannot use this command to modify the port used by an existing LS; the *port_name* specified on the command must match the previous definition of the LS. The LS can be modified only if it is not started.

Supplied Parameters

Parameter name [define_qllc_ls]	Type	Length	Default
ls_name	character	8	
description	character	31	(null string)
port_name	character	8	
adj_cp_name	character	17	(null string)
adj_cp_type	constant		LEARN_NODE
address	character	15	(null string)
auto_act_supp	constant		NO
tg_number	decimal		0
limited_resource	constant		NO
solicit_sscp_sessions	constant		NO
pu_name	character	8	(taken from ls_name)
disable_remote_act	constant		NO
dspu_services	constant		NONE
dspu_name	character	8	(taken from ls_name)
dlus_name	character	17	(null string)
bkup_dlus_name	character	17	(null string)
hpr_supported	constant		NO
link_deact_timer	decimal		30
default_nn_server	constant		NO
ls_attributes	constant		SNA
adj_node_id	hex array	4	(0x0)
local_node_id	hex array	4	(0x0)
cp_cp_sess_support	constant		YES
use_default_tg_chars	constant		YES
effect_cap	decimal		64000
connect_cost	decimal		0
byte_cost	decimal		0
security	constant		SEC_PUBLIC_SWITCHED_NETWORK
prop_delay	constant		PROP_DELAY_PKT_SWITCHED_NET
user_def_parm_1	decimal		128
user_def_parm_2	decimal		128
user_def_parm_3	decimal		128
target_pacing_count	decimal		7
max_ifrm_rcvd	decimal		0
dlus_retry_timeout	decimal		0
dlus_retry_limit	decimal		0
conventional_lu_compression	constant		NO
branch_link_type	constant		NONE
max_send_btu_size	decimal		265
ls_role	constant		USE_PORT_DEFAULTS
initially_active	constant		NO
react_timer	decimal		30
react_timer_retry	decimal		65535
restart_on_normal_deact	constant		NO
vc_type	constant		SVC
fac	hex array	128	(null string)
pvc_id	decimal		1
cud	hex array	128	C3
dddlu_offline_supported	constant		NO

Supplied parameters are:

ls_name
: Name of the link station to be defined.

description
: A text string describing the LS. Communications Server for Linux uses this

string for information only. It is stored in the node's configuration file and returned on the **query_ls**, **query_pu**, and **query_downstream_pu** commands.

port_name
Name of port associated with this link station. This name must match the name of a defined port.

adj_cp_name
Fully qualified name of the adjacent CP for this LS. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character CP name. This parameter is used in the following ways:

- If the *adj_cp_type* parameter is set to NETWORK_NODE or END_NODE, and preassigned TG numbers are being used, set this parameter to the CP name defined at the adjacent node; if the adjacent node sends a CP name during XID exchange, it will be checked against this value.
- If *adj_cp_type* is set to BACK_LEVEL_LEN_NODE, Communications Server for Linux uses this value only as an identifier; set it to any string that does not match other CP names defined at this node.
- If *adj_cp_type* is set to any other value, or if preassigned TG numbers are not being used, there is no need to specify this parameter; Communications Server for Linux checks the CP name only if one is specified.

adj_cp_type
Adjacent node type.

If the adjacent node is an APPN node and preassigned TG numbers are not being used, this parameter is usually set to LEARN_NODE, indicating that the node type is unknown; Communications Server for Linux will determine the type during XID exchange.

If preassigned TG numbers are being used, you must specify the node type explicitly. You can also specify the type as an additional security check if preassigned TG numbers are not being used. In this case, Communications Server for Linux will reject a connection attempt from the adjacent node if its node type does not match the one specified for this parameter. Possible values are:

LEARN_NODE
The adjacent node type is unknown and Communications Server for Linux determines the type during XID exchange.

END_NODE
The adjacent node is an End Node, a Branch Network Node acting as an End Node from the local node's perspective, or a LEN node that includes the Network Name CV in its XID3.

NETWORK_NODE
The adjacent node is a Network Node, or a Branch Network Node acting as a Network Node from the local node's perspective.

If the adjacent node is not an APPN node, possible values are:

BACK_LEVEL_LEN_NODE
The adjacent node is one that does not include the Network Name control vector in its XID3.

HOST_XID3
The adjacent node is a host node; Communications Server for Linux responds to a polling XID from the node with a format 3 XID.

HOST_XID0
The adjacent node is a host node; Communications Server for Linux responds to a polling XID from the node with a format 0 XID.

DSPU_XID
The adjacent node is a downstream PU; Communications Server for Linux includes XID exchange in link activation. The *dspu_name* and *dspu_services* parameters must also be set.

DSPU_NOXID
The adjacent node is a downstream PU; Communications Server for Linux does not include XID exchange in link activation. The *dspu_name* and *dspu_services* parameters must also be set.

If you want to run independent LU 6.2 traffic over this LS, you must set the *adj_cp_type* parameter to `LEARN_NODE`, `END_NODE`, `NETWORK_NODE`, or `BACK_LEVEL_LEN_NODE`.

address Destination address of the remote link station.

This parameter is used only for SVC outgoing calls (defined by the *vc_type* parameter and by the link activation limit parameters on **define_qllc_port**); it is ignored for incoming calls or for PVC.

The address is a string of 1–15 characters in X.25 (1980) format; later address formats are not supported.

auto_act_supp
Specifies whether the link can be automatically activated when required by a session. Possible values are:

YES The link can be automatically activated.

The reactivation timer parameters are ignored. If the LS fails, Communications Server for Linux does not attempt to reactivate it until a dependent LU application that was using the session attempts to restart the session; an LS used by independent LUs is not reactivated by Communications Server for Linux and must be manually restarted.

The following restrictions also apply:

- If the link is to an APPN node, the LS must have a preassigned TG number defined (see the *tg_number* parameter), and *cp_cp_sess_support* must be set to `NO`.
- If either the local node or the adjacent node is an end node, the LS must also be defined to automatically activate at the adjacent node.

NO The link cannot be automatically activated.

tg_number
Preassigned TG number. This parameter is used only if the adjacent node is an APPN node (*adj_cp_type* is either `NETWORK_NODE` or `END_NODE`); it is ignored otherwise.

This TG number is used to represent the link when the link is activated. The node does not accept any other number from the adjacent node during activation of this link; if the adjacent node is using preassigned TG numbers, the same TG number must be defined by the adjacent node on the adjacent link station.

If the local node is a LEN node, or if the adjacent node is a LEN node and the link is defined to automatically activate, set the TG number to 1. Otherwise, specify a number in the range 1–20, or 0 (zero) to indicate that the TG number is not preassigned and is negotiated when the link is activated.

If a preassigned TG number is defined, the *adj_cp_name* parameter must also be defined, and the *adj_cp_type* parameter must be set to either END_NODE or NETWORK_NODE.

limited_resource
Specifies whether this link station is a limited resource and is automatically deactivated when there are no sessions using the link. Link stations on a nonswitched port cannot be configured as limited resource. Possible values are:

NO The link is not a limited resource and is not automatically deactivated.

NO_SESSIONS
The link is a limited resource and is automatically deactivated when no active sessions are using it.

INACTIVITY
The link is a limited resource and is automatically deactivated when no active sessions are using it, or when no data has flowed on the link for the time period specified by the *link_deact_timer* parameter.

- If no SSCP-PU session is active across the link, the node deactivates the link immediately.
- If an SSCP-PU session is active but no traffic has flowed for the specified time period, the node sends REQDISCONT(normal) to the host. The host is then responsible for deactivating all LUs and the PU, at which time the local node will deactivate the link. However, the host may not deactivate LUs with active PLU-SLU sessions; in this case, the link remains active until all these sessions are deactivated (for example by the user logging out). This behavior can be changed by using options in the *ptf* parameter of the **start_node** command.

A limited resource link station can be configured for CP-CP session support by setting this parameter to NO_SESSIONS and *cp_cp_sess_support* to YES. In this case, if CP-CP sessions are brought up over the link, Communications Server for Linux does not treat the link as a limited resource (and therefore does not deactivate it).

solicit_sscp_sessions
Specifies whether to request the adjacent node to initiate sessions between the SSCP and the local CP and dependent LUs. This parameter is used only if the adjacent node is an APPN node (*adj_cp_type* is either NETWORK_NODE or END_NODE); it is ignored otherwise. If the adjacent node is a host (*adj_cp_type* is either HOST_XID3 or HOST_XID0), Communications Server for Linux always requests the host to initiate SSCP sessions.

Possible values are:

YES Request the adjacent node to initiate SSCP sessions.

NO Do not request the adjacent node to initiate SSCP sessions.

If the adjacent node is an APPN node and *dspu_services* is set to a value other than NONE, this parameter must be set to NO.

pu_name

Name of the local PU that uses this link. This parameter is required only if *adj_cp_type* is set to HOST_XID3 or HOST_XID0, or if *solicit_sscp_sessions* is set to YES; it is ignored otherwise. This name is a type-A character string starting with a letter.

You cannot change the PU name on an LS that is already defined.

If the PU name is required and you do not specify it, the default is the same as the LS name. To ensure that this name is a valid type-A character string, Communications Server for Linux converts it to uppercase; if the string begins with a numeric character, this character is either removed or preceded by the characters "PU."

disable_remote_act

Specifies whether to prevent activation of the LS by the remote node. Possible values are:

YES The LS can be activated only by the local node; if the remote node attempts to activate it, Communications Server for Linux will reject the attempt.

NO The LS can be activated by the remote node.

dspu_services

Specifies the services that the local node provides to the downstream PU across this link. This parameter is used only if the *adj_cp_type* parameter is set to DSPU_XID or DSPU_NOXID or if the *solicit_sscp_sessions* parameter is set to NO; it is reserved otherwise. Possible values are:

PU_CONCENTRATION
Local node provides SNA gateway for the downstream PU. The local node must be defined to support SNA gateway.

DLUR Local node provides DLUR services for the downstream PU. The local node must be defined to support DLUR. (DLUR is not supported on end node.)

NONE Local node provides no services for the downstream PU.

dspu_name

Name of the downstream PU. The name is a type-A character string starting with a letter. To ensure that this name is a valid type-A character string, Communications Server for Linux converts it to uppercase; if the string begins with a numeric character, this character is either removed or preceded by the characters "PU."

This parameter is reserved except when both of the following conditions are true:

- The *solicit_sscp_sessions* parameter is set to NO
- The *dspu_services* parameter is set to PU_CONCENTRATION or DLUR

If both of these conditions are true and you do not specify a value for *dspu_name*, the default is the same as the LS name.

If the downstream PU is used for DLUR, this name should match the PU name configured on the host. (Communications Server for Linux sends both the PU name and PU ID to the host to identify the PU. The host normally identifies the PU by its PU name, or by the PU ID if it cannot find a matching PU name.)

dlus_name
Name of the DLUS node from which DLUR solicits SSCP services when the link to the downstream node is activated. This parameter is reserved if *dspu_services* is not set to `DLUR`.

Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character LU name.

To specify the global default DLUS defined using the **define_dlur_defaults** command, do not specify this parameter. If this parameter is not specified and there is no global default DLUS, then DLUR will not initiate SSCP contact when the link is activated.

bkup_dlus_name
Name of the backup DLUS node from which DLUR solicits SSCP services if the node specified by *dlus_name* is not active. This parameter is reserved if *dspu_services* is not set to `DLUR`.

Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character backup LU name.

To specify the global backup default DLUS, defined using **define_dlur_defaults**, do not specify this parameter.

hpr_supported
Specifies whether HPR is supported on this link. This parameter must be set to NO unless the *adj_cp_type* parameter indicates that the link connects to an APPN node. Possible values are:

YES HPR is supported on this link.

NO HPR is not supported on this link.

link_deact_timer
Limited resource link deactivation timer, in seconds. A limited resource link is automatically deactivated if no data flows over the link for the time specified by this parameter. This parameter is not used if *limited_resource* is set to any value other than `INACTIVITY`.

The minimum value is `5`; values in the range `1`–`4` will be interpreted as `5`.

The value `0` (zero) indicates one of the following:

- If the *hpr_supported* parameter is set to `YES`, the default deactivation timer value of `30` is used.
- If the *hpr_supported* parameter is set to `NO`, no timeout is used (the link is not deactivated, as if *limited_resource* were set to `NO`).

default_nn_server
For an end node this parameter specifies whether the link handled by this link station is a link supporting CP-CP sessions to a network node that can act as the local node's network node server. When the local node has no CP-CP sessions to a network node server and needs to establish them, the local node checks this parameter on its defined link stations to find a suitable LS to activate. This enables you to specify which adjacent NNs are suitable to act as the NN server (for example, to avoid using NNs that are accessed by expensive or slow links).

Possible values are:

YES This link supports CP-CP sessions to a network node that can act as the local node's NN server; the local node can automatically activate this link if it needs to contact an NN server. The *cp_cp_sess_support* parameter must be set to YES.

NO This link should not be automatically activated when an attempt is made to contact an NN server.

If the local node is not an end node, this parameter is ignored. If the local node is not an end node, this parameter is ignored.

ls_attributes
Attributes of the remote system with which Communications Server for Linux is communicating.

Specify SNA unless you are communicating with a host of another type. Possible values are:

SNA Standard SNA host.

FNA Fujitsu Network Architecture (VTAM-F) host

HNA Hitachi Network Architecture host

SUPPRESS_CP_NAME
Suppress the CP name associated with the remote node. Use a + character to combine this value with SNA, FNA, or HNA.

If *adj_cp_type* is set to BACK_LEVEL_LEN_NODE, and the remote LEN node associated with this LS cannot accept the Network Name CV in the format 3 XID it receives, use a + character to combine the value SNA, FNA, or HNA with SUPPRESS_CP_NAME (for example, SNA+SUPPRESS_CP_NAME).

If *adj_cp_type* is set to any other value, the SUPPRESS_CP_NAME option is ignored.

adj_node_id
Node ID of adjacent node. This ID is a 4-byte hexadecimal string, consisting of a block number (three hexadecimal digits) and a node number (five hexadecimal digits). To disable node ID checking, do not specify this parameter.

local_node_id
Node ID sent in XIDs on this LS. This ID is a 4-byte hexadecimal string, consisting of a block number (three hexadecimal digits) and a node number (five hexadecimal digits). To use the node ID specified in **define_node**, do not specify this parameter.

cp_cp_sess_support
Specifies whether CP-CP sessions are supported. This parameter is valid only if the adjacent node is an end node or a network node (*adj_cp_type* is NETWORK_NODE, END_NODE, or LEARN_NODE); it is ignored otherwise. If both the local node and the adjacent node are network nodes, this parameter should be set to YES in order to use APPN functions between these nodes.

Possible values are:

YES CP-CP sessions are supported.

NO CP-CP sessions are not supported.

use_default_tg_chars
Specifies whether to use the default TG characteristics supplied on

define_qllc_port. The TG characteristics apply only if the link is to an APPN node; this parameter, and *effect_cap* through *user_def_parm_3* parameters are ignored otherwise. Possible values are:

YES Use the default TG characteristics; ignore *effect_cap* through *user_def_parm_3* parameters on this command.

NO Use *effect_cap* through *user_def_parm_3* parameters on this command.

effect_cap
A decimal value representing the line speed in bits per second.

connect_cost
Cost per connect time. Valid values are integer values in the range `0–255`, where `0` is the lowest cost per connect time and `255` is the highest cost per connect time.

byte_cost
Cost per byte. Valid values are integer values in the range `0–255`, where `0` is the lowest cost per byte and `255` is the highest cost per byte.

security
Security level of the network. Possible values are:

SEC_NONSECURE
No security.

SEC_PUBLIC_SWITCHED_NETWORK
Data is transmitted over a public switched network.

SEC_UNDERGROUND_CABLE
Data is transmitted over secure underground cable.

SEC_SECURE_CONDUIT
Data is transmitted over a line in a secure conduit that is not guarded.

SEC_GUARDED_CONDUIT
Data is transmitted over a line in a conduit that is protected against physical tapping.

SEC_ENCRYPTED
Data is encrypted before transmission over the line.

SEC_GUARDED_RADIATION
Data is transmitted over a line that is protected against physical and radiation tapping.

prop_delay
Propagation delay (the time that a signal takes to travel the length of the link). Specify one of the following values, according to the type of link:

PROP_DELAY_MINIMUM
Minimum propagation delay.

PROP_DELAY_LAN
Delay is less than .5 microseconds (typical for a LAN).

PROP_DELAY_TELEPHONE
Delay is in the range .5–50 microseconds (typical for a telephone network).

PROP_DELAY_PKT_SWITCHED_NET
Delay is in the range 50–250 microseconds (typical for a packet-switched network).

PROP_DELAY_SATELLITE
Delay is greater than 250 microseconds (typical for a satellite link).

PROP_DELAY_MAXIMUM
Maximum propagation delay.

user_def_parm_1 **through** *user_def_parm_3*
User-defined parameters that you can use to include TG characteristics not covered by the previously described parameters. Each of these parameters must be set to a value in the range 0–255.

target_pacing_count
Indicates the desired pacing window size. Specify a value in the range 1–32,767.

max_ifrm_rcvd
The maximum number of I-frames that can be received by this link station before an acknowledgment is sent. Specify a value in the range 0–127. If 0 is specified, the value from **define_qllc_port** is used.

dlus_retry_timeout
The interval in seconds between the second and subsequent attempts to contact the DLUS specified by the *dlus_name* and *bkup_dlus_name* parameters. The interval between the first and second attempts is always 1 second.

Specify a value in the range 0– 65,535. If you specify 0, the default specified using **define_dlur_defaults** is used. This parameter is ignored if the *dspu_services* parameter is not set to DLUR.

dlus_retry_limit
Retry count for contacting a DLUS. This parameter is used to specify the number of times Communications Server for Linux should retry if it fails to contact the DLUS on the first attempt.

Specify a value in the range 1–65,534, or specify 65,535 to indicate that Communications Server for Linux should retry indefinitely until it contacts the DLUS.

conventional_lu_compression
Specifies whether data compression is requested for LU 0–3 sessions on this link. This parameter is used only if this link carries LU 0–3 traffic; it does not apply to LU 6.2 sessions. Possible values are:

YES Data compression should be used for LU 0–3 sessions on this link if the host requests it.

NO Data compression should not be used for LU 0–3 sessions on this link.

branch_link_type
This parameter applies only if the local node is a Branch Network Node; it is not used if the local node is any other type.

If the parameter *adj_cp_type* is set to NETWORK_NODE, END_NODE, APPN_NODE, or BACK_LEVEL_LEN_NODE, this parameter defines whether the link is an uplink or a downlink. Possible values are:

UPLINK The link is an uplink.

DOWNLINK
The link is a downlink.

If *adj_cp_type* is set to NETWORK_NODE, this parameter must be set to UPLINK.

adj_brnn_cp_support
This parameter applies only if the local node is a Branch Network Node and the adjacent node is a network node (the parameter *adj_cp_type* is set to NETWORK_NODE, or it is set to APPN_NODE and the node type discovered during XID exchange is network node). It is not used if the local and remote nodes are any other type.

This parameter defines whether the adjacent node can be a Branch Network Node that is acting as a Network Node from the point of view of the local node. Possible values are:

ALLOWED
The adjacent node is allowed (but not required) to be a Branch Network Node.

REQUIRED
The adjacent node must be a Branch Network Node.

PROHIBITED
The adjacent node must not be a Branch Network Node.

If *adj_cp_type* is set to NETWORK_NODE and *auto_act_supp* is set to YES, this parameter must be set to REQUIRED or PROHIBITED.

max_send_btu_size
Maximum BTU size that can be sent from this link station. This value is used to negotiate the maximum BTU size that a pair of link stations can use to communicate with each other. The value includes the length of the TH and RH (total 9 bytes), as well as the RU. Specify a value in the range 265–65,535.

ls_role Link station role. This parameter is usually set to USE_PORT_DEFAULTS, specifying that the LS role is to be taken from the definition of the port that owns this LS.

If you need to override the port's LS role for an individual LS, specify one of the following values:

LS_PRI Primary

LS_SEC Secondary

LS_NEG Negotiable

initially_active
Specifies whether this LS is automatically started when the node is started. Possible values are:

YES The LS is automatically started when the node is started.

NO The LS is not automatically started; it must be manually started.

If the LS is a PVC link, you are recommended to set this parameter to YES to ensure that the link is always available.

react_timer
Reactivation timer for reactivating a failed LS. This parameter specifies the amount of time (in seconds) that Communications Server for Linux should wait before retrying to activate the LS that failed. If the *react_timer_retry*

parameter is a nonzero value, then Communications Server for Linux should retry activating the LS if it fails. If *react_timer_retry* parameter value is zero, this parameter is ignored.

react_timer_retry
Retry count for reactivating a failed LS. This parameter is used to specify whether Communications Server for Linux attempts to reactivate the LS if it fails while in use (or if an attempt to start the LS fails).

Specify `0` (zero) to indicate that Communications Server for Linux should not attempt to reactivate the LS. Specify the number of retries to be made to indicate that Communications Server for Linux should attempt to reactivate the LS. A value of `65,535` indicates that Communications Server for Linux should retry indefinitely until the LS is reactivated.

Communications Server for Linux waits for the time specified by the *react_timer* parameter between successive retries. If the LS is not successfully reactivated by the end of the retry count, or if **stop_ls** is issued while Communications Server for Linux is retrying the activation, no further retries are made; the LS remains inactive unless **start_ls** is issued for it.

If the *auto_act_supp* parameter is set to `YES`, the *react_timer* and *react_timer_retry* parameters are ignored; if the link fails, Communications Server for Linux does not attempt to reactivate it until the user application that was using the session attempts to restart the session.

If the LS is a PVC link, you are recommended to set this parameter to a non-zero value to ensure that the link is always available.

restart_on_normal_deact
Specifies whether Communications Server for Linux should attempt to reactivate the LS if it is deactivated normally by the remote system. Possible values are:

YES If the remote system deactivates the LS normally, Communications Server for Linux attempts to reactivate it, using the same retry timer and count values as for reactivating a failed LS (the *react_timer* and *react_timer_retry* parameters above).

NO If the remote system deactivates the LS normally, Communications Server for Linux does not attempt to reactivate it.

If the LS is a host link (specified by the *adj_cp_type* parameter), or is automatically started when the node is started (the *initially_active* parameter is set to `YES`), this parameter is ignored; Communications Server for Linux always attempts to reactivate the LS if it is deactivated normally by the remote system (unless *react_timer_retry* is zero).

vc_type
The virtual circuit type of the LS. Possible values are:

SVC Switched virtual circuit

PVC Permanent virtual circuit

fac Specify any facilities data required in the call packet sent to the remote system. Check with the administrator of your X.25 network, or the administrator of the remote system, to determine what to specify in this parameter.

pvc_id PVC identifier. Set this parameter to a decimal number to identify which

PVC (from the range of PVCs defined for your X.25 provider software) to use for this LS. This parameter is reserved if *vc_type* is set to SVC.

cud Call user data. This parameter identifies the protocol to use over the underlying X.25 virtual circuit, and is used only if the *vc_type* parameter is set to SVC.

For most implementations, this parameter is set to a single hex byte, which is 0xC3 to request that the called node supports the 1980 QLLC level, or 0xCB to request 1984 support. Some remote systems may require additional bytes; contact the System Administrator of the remote system if necessary.

dddlu_offline_supported
Specifies whether the local PU should send NMVT (power off) messages to the host. If the host system supports DDDLU (Dynamic Definition of Dependent LUs), Communications Server for Linux sends NMVT (power off) to the host when it has finished using a dynamically defined LU. This allows the host to save resources by removing the definition when it is no longer required.

This parameter is used only if this link is to a host (*solicit_sscp_sessions* is set to YES and *dspu_services* is not set to NONE).

Possible values are:

YES The local PU sends NMVT (power off) messages to the host.

NO The local PU does not send NMVT (power off) messages to the host.

If the host supports DDDLU but does not support the NMVT (power off) message, this parameter must be set to NO.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

CANT_MODIFY_PORT_NAME
The *ls_name* parameter matched the name of an existing LS, but the *port_name* parameter did not match the existing definition. You cannot modify the port name when changing the definition of an existing LS.

DEF_LINK_INVALID_SECURITY
The *security* parameter was not set to a valid value.

INVALID_AUTO_ACT_SUPP
The *auto_act_supp* parameter was not set to a valid value or was set to YES when *cp_cp_sess_support* was also set to YES.

INVALID_CP_NAME
The *adj_cp_name* parameter contained a character that was not valid, was not in the correct format, or was not specified when required.

INVALID_LIMITED_RESOURCE
The *limited_resource* parameter was not set to a valid value.

INVALID_LINK_NAME
The *ls_name* parameter contained a character that was not valid.

INVALID_NODE_TYPE
The *adj_cp_type* parameter was not set to a valid value.

INVALID_PORT_NAME
The *port_name* parameter did not match the name of any defined port.

INVALID_PU_NAME
The *pu_name* parameter did not match the name of any defined PU, or was set to a new value on an already-defined LS.

INVALID_DSPU_NAME
The *dspu_name* parameter did not match the name of any defined PU or was set to a new value on an already-defined LS.

INVALID_DSPU_SERVICES
The *dspu_services* parameter was not set to a valid value or was set when not expected.

INVALID_SOLICIT_SSCP_SESS
The *solicit_sscp_sess* parameter was not set to a valid value.

INVALID_TARGET_PACING_CNT
The *target_pacing_count* parameter was not set to a valid value.

INVALID_DLUS_NAME
The *dlus_name* parameter contained a character that was not valid or was not in the correct format.

INVALID_BKUP_DLUS_NAME
The *bkup_dlus_name* parameter contained a character that was not valid or was not in the correct format.

HPR_NOT_SUPPORTED
A reserved parameter was set to a nonzero value.

INVALID_TG_NUMBER
The TG number supplied was not in the valid range.

MISSING_CP_NAME
A TG number was defined, but no CP name was supplied.

MISSING_CP_TYPE
A TG number was defined, but no CP type was supplied.

MISSING_TG_NUMBER
The link was defined as automatically activated, but no TG number was supplied.

PARALLEL_TGS_NOT_SUPPORTED
This node cannot support more than one LS defined between it and the same adjacent node.

INVALID_DLUS_RETRY_LIMIT
The value specified for *dlus_retry_limit* was not valid.

INVALID_DLUS_RETRY_TIMEOUT
The value specified for *dlus_retry_timeout* was not valid.

INVALID_LS_ROLE
The value specified for the *ls_role* parameter is not valid.

INVALID_NODE_TYPE_FOR_HPR
The node type specified for the *adj_cp_type* parameter does not support HPR.

INVALID_BTU_SIZE
The value specified for the *max_send_btu_size* parameter was not valid.

INVALID_MAX_IFRM_RCVD
The value specified for the *max_ifrm_rcvd* parameter was not valid.

INVALID_BRANCH_LINK_TYPE
The *branch_link_type* parameter was not set to a valid value.

INVALID_BRNN_SUPPORT
The *adj_brnn_cp_support* parameter was not set to a valid value.

BRNN_SUPPORT_MISSING
The *adj_brnn_cp_support* parameter was set to `ALLOWED`; this value is not valid because the adjacent node is a Network Node and *auto_act_supp* is set to `YES`.

INVALID_UPLINK
The *branch_link_type* parameter was set to `UPLINK`, but the definition of an existing LS between the local and adjacent nodes specifies that it is a downlink. The branch link type must be the same for all LSs between the same two nodes.

INVALID_DOWNLINK
The *branch_link_type* parameter was set to `DOWNLINK`, but the definition of an existing LS between the local and adjacent nodes specifies that it is an uplink. The branch link type must be the same for all LSs between the same two nodes.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
`STATE_CHECK`

secondary_rc
Possible values are:

DUPLICATE_CP_NAME
A link to the CP name specified in the *adj_cp_name* parameter has already been defined.

DUPLICATE_DEST_ADDR
A link to the destination address specified in the *address* parameter has already been defined.

INVALID_LINK_NAME
The link station value specified in the *ls_name* parameter was not valid.

INVALID_NUM_LS_SPECIFIED
The number of link stations specified was not valid.

LOCAL_CP_NAME
The name specified for the *adj_cp_name* parameter is identical to the local CP name.

LS_ACTIVE
The link station specified in the *ls_name* parameter is currently active.

PU_ALREADY_DEFINED
The PU specified in the *pu_name* parameter has already been defined.

DSPU_ALREADY_DEFINED
The downstream PU specified in the *dspu_name* parameter has already been defined.

DSPU_SERVICES_NOT_SUPPORTED
The *dspu_services* parameter was used to request a service that is not supported.

DEPENDENT_LU_NOT_SUPPORTED
The *solicit_sscp_sessions* parameter was set to YES, but dependent LU is not supported.

DUPLICATE_TG_NUMBER
The TG number specified in the *tg_number* parameter has already been defined.

TG_NUMBER_IN_USE
The TG number specified for the *tg_number* parameter is already being used by another LS.

define_qllc_port

The **define_qllc_port** command is used to define a new QLLC port or modify an existing QLLC port. Before issuing this command, you must define the DLC that this port uses.

You can modify an existing port only if it is not started. You cannot change the DLC used by an existing port; the *dlc_name* specified when modifying an existing port must match the DLC that was specified on the initial definition of the port.

For information about defining a port that will accept incoming calls, see "Incoming Calls" on page 157.

Supplied Parameters

Parameter name	Type	Length	Default
[define_qllc_port]			
port_name	character	8	
description	character	31	(null string)
dlc_name	character	8	
max_rcv_btu_size	decimal		265
tot_link_act_lim	decimal		64
inb_link_act_lim	decimal		0
out_link_act_lim	decimal		0
ls_role	constant		LS_NEG
implicit_dspu_template	character	8	(null string)
implicit_ls_limit	decimal		0
implicit_dspu_services	constant		NONE
implicit_deact_timer	decimal		30
act_xid_exchange_limit	decimal		9
nonact_xid_exchange_limit	decimal		5
ls_xmit_rcv_cap	constant		LS_TWS
max_ifrm_rcvd	decimal		7
target_pacing_count	decimal		7
max_send_btu_size	decimal		265
address	character	15	(null string)
implicit_cp_cp_sess_support	constant		YES
implicit_limited_resource	constant		NO
implicit_hpr_support	constant		NO
implicit_uplink_to_en	constant		NO
effect_cap	decimal		64000
connect_cost	decimal		0
byte_cost	decimal		0
security	constant		SEC_PUBLIC_SWITCHED_NETWORK
prop_delay	constant		PROP_DELAY_PKT_SWITCHED_NET
user_def_parm_1	decimal		128
user_def_parm_2	decimal		128
user_def_parm_3	decimal		128
initially_active	constant		YES
cud_mode	constant		DONTCARE
cud_match	hex array	128	(null string)
add_mode	constant		DONTCARE
add_len	decimal		0

Supplied parameters are:

port_name
: Name of the port to be defined. This name is a character string using any locally displayable characters.

description
: A text string describing the port. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_port** command.

dlc_name
: Name of the associated DLC. This name is a character string using any locally displayable characters. The specified DLC must have already been defined.

max_rcv_btu_size
: Maximum BTU size that can be received. This value includes the length of the TH and RH (total 9 bytes) as well as the RU. Specify a value in the range 265–65,535.

tot_link_act_lim
Total link activation limit—the maximum number of links that can be active at any time using this port.

inb_link_act_lim
Inbound link activation limit—the number of links reserved for inbound activation. The sum of *inb_link_act_lim* and *out_link_act_lim* must not exceed *tot_link_act_lim*; the difference between *inb_link_act_lim* and *tot_link_act_lim* defines the maximum number of links that can be activated outbound at any time.

out_link_act_lim
Outbound link activation limit—the number of links reserved for outbound activation. The sum of *inb_link_act_lim* and *out_link_act_lim* must not exceed *tot_link_act_lim*; the difference between *out_link_act_lim* and *tot_link_act_lim* defines the maximum number of links that can be activated inbound at any time.

ls_role Link station role. Possible values are:

LS_PRI Primary

LS_SEC Secondary

LS_NEG Negotiable

implicit_dspu_template
Specifies the DSPU template, defined on the **define_dspu_template** command. This template is used for definitions if the local node is to provide SNA gateway for an implicit link activated on this port. If the template specified does not exist or is already at its instance limit when the link is activated, activation will fail. This template name is an 8-byte string in a locally displayable character set.

If the *implicit_dspu_services* parameter is not set to `PU_CONCENTRATION`, the *implicit_dspu_template* parameter is reserved.

implicit_ls_limit
Specifies the maximum number of implicit link stations which can be active on this port simultaneously, including dynamic links and links activated for Discovery. Specify a value in the range `1`–`65,534` or specify `0` (zero) to indicate no limit. A value of `NO_IMPLICIT_LINKS` indicates that no implicit links are allowed.

implicit_dspu_services
Specifies the services that the local node will provide to the downstream PU across implicit links activated on this port. Possible values are:

DLUR Local node will provide DLUR services for the downstream PU (using the default DLUS configured through the **define_dlur_defaults** command).

PU_CONCENTRATION
Local node will provide SNA gateway for the downstream PU. It will also put in place definitions as specified by the DSPU template specified for the parameter *implicit_dspu_template*.

NONE Local node will provide no services for this downstream PU.

implicit_deact_timer
Implicit limited resource link deactivation timer, in seconds.

If *implicit_hpr_support* is set to YES and *implicit_limited_resource* is set to NO_SESSIONS, an implicit link is automatically deactivated if no data flows on it for the time specified by this parameter and no sessions are using the link.

If *implicit_limited_resource* is set to INACTIVITY, an implicit link using this port is automatically deactivated if no data flows on it for the time specified by this parameter.

The minimum value is 5; values in the range 1–4 will be interpreted as 5. The value 0 (zero) indicates no timeout (the link is not deactivated, as though *implicit_limited_resource* were set to NO). This parameter is reserved if *implicit_limited_resource* is set to NO.

implicit_uplink_to_en
This parameter applies only if the local node is a Branch Network Node; it is reserved if the local node is any other type.

If the adjacent node is an end node, this parameter specifies whether implicit link stations off this port are uplink or downlink. This parameter is ignored if there are existing links to the same adjacent node, because in this case the existing links are used to determine the link type. Possible values are:

YES Implicit links to an End Node are uplinks.

NO Implicit links to an End Node are downlinks.

act_xid_exchange_limit
Activation XID exchange limit. Specify a value in the range 0–65,535.

nonact_xid_exchange_limit
Nonactivation XID exchange limit. Specify a value in the range 0–65,535.

ls_xmit_rcv_cap
Specifies the link station transmit/receive capability. Possible values are:

LS_TWS Two-way simultaneous

LS_TWA Two-way alternating

max_ifrm_rcvd
Maximum number of I-frames that can be received by the local link stations before an acknowledgment is sent. Specify a value in the range 1–127.

target_pacing_count
Indicates the desired pacing window size. Specify a value in the range 1–32,767.

max_send_btu_size
Maximum BTU size that can be sent from this port. This value is used to negotiate the maximum BTU size that a pair of link stations can use to communicate with each other. The value includes the length of the TH and RH (total 9 bytes), as well as the RU.

address Local X.25 DTE address of the port.

implicit_cp_cp_sess_support
Specifies whether CP-CP sessions are allowed for implicit link stations using this port. Possible values are:

YES CP-CP sessions are allowed for implicit link stations.

NO CP-CP sessions are not allowed for implicit link stations.

implicit_limited_resource
Specifies whether implicit link stations off this port are defined as limited resources. Possible values are:

NO Implicit links are not limited resources and are not automatically deactivated.

NO_SESSIONS
Implicit links are limited resources and are automatically deactivated when no active sessions are using them.

INACTIVITY
Implicit links are limited resources and are automatically deactivated when no active sessions are using them or when no data has flowed for the time period specified by the *implicit_deact_timer* parameter.

- If no SSCP-PU session is active across the link, the node deactivates the link immediately.
- If an SSCP-PU session is active but no traffic has flowed for the specified time period, the node sends REQDISCONT(normal) to the host. The host is then responsible for deactivating all LUs and the PU, at which time the local node will deactivate the link. However, the host may not deactivate LUs with active PLU-SLU sessions; in this case, the link remains active until all these sessions are deactivated (for example by the user logging out). This behavior can be changed by using options in the *ptf* parameter of the **start_node** command.

implicit_hpr_support
Specifies whether High Performance Routing (HPR) is supported on implicit links. Possible values are:

YES HPR is supported on implicit links.

NO HPR is not supported on implicit links.

effect_cap **through** *user_def_parm_3*
Default TG characteristics used for implicit link stations using this port. These characteristics are also used as the default TG characteristics for defined link stations that do not have explicitly defined TG characteristics. The TG characteristics parameters are ignored if the LS is to a downstream PU.

For more information about these parameters, see "define_tr_ls, define_ethernet_ls" on page 209.

initially_active
Specifies whether this port is automatically started when the node is started. Possible values are:

YES The port is automatically started when the node is started.

NO The port is automatically started only if an LS that uses the port is defined as initially active; otherwise, it must be manually started.

cud_mode
Specifies the type of matching required between the call user data (CUD) supplied on an incoming call and the *cud_match* parameter. Possible values are:

DONTCARE
CUD on incoming calls is not checked.

IDENTITY
The received CUD must match the string specified in the *cud_match* parameter.

STARTSWITH
The string specified in the *cud_match* parameter is matched to the same number of initial bytes of the received CUD; any bytes following the initial bytes are not checked.

cud_match
Call user data to be used for verifying incoming calls. If *cud_mode* is set to IDENTITY or STARTSWITH, incoming calls are accepted only if they specify a CUD string that matches the value defined in the *cud_match* parameter. If *cud_mode* is set to DONTCARE, the *cud_match* parameter is ignored and CUD strings on incoming calls are not checked.

add_mode
Specifies the type of matching required between the address supplied on an incoming call and the port address defined in the *address* parameter. Possible values are:

DONTCARE
The address on incoming calls is not checked.

IDENTITY
The received address must match the string specified in the *address* parameter.

STARTSWITH
The initial bytes (the number of bytes is specified in the *add_len* parameter) of the received address must match the string specified in the address parameter; any bytes greater than the number specified in *add_len* are not checked.

If the *address* parameter is not specified, this parameter must be set to DONTCARE.

add_len
If *add_mode* is set to STARTSWITH, this parameter specifies the number of bytes of the port address to be checked.

For example, if *add_len* is set to 2, an incoming call is accepted if the first two bytes of the address supplied on the call match the first two bytes of the *address* parameter (regardless of whether subsequent bytes match).

For other values of *add_mode*, this parameter is ignored.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_PORT_NAME
The value specified in the *port_name* parameter was not valid.

INVALID_DLC_NAME
The specified *dlc_name* did not match any defined DLC.

INVALID_BTU_SIZE
The *max_rcv_btu_size* parameter was not set to a valid value.

INVALID_LINK_ACTIVE_LIMIT
One of the activation limit parameters, *inb_link_act_lim*, *out_link_act_lim*, or *tot_link_act_lim*, was not set to a valid value.

INVALID_MAX_IFRM_RCVD
The *max_ifrm_rcvd* parameter was not set to a valid value.

HPR_NOT_SUPPORTED
A reserved parameter was set to a nonzero value.

INVALID_LS_ROLE
The *ls_role* parameter was not set to a valid value.

INVALID_DSPU_SERVICES
The *implicit_dspu_services* parameter was not set to a valid value.

INVALID_TEMPLATE_NAME
The DSPU template specified on the *implicit_dspu_template* parameter was not valid.

INVALID_IMPLICIT_UPLINK
The *implicit_uplink_to_en* parameter was not set to a valid value.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

PORT_ACTIVE
The specified port cannot be modified because it is currently active.

DUPLICATE_PORT_NUMBER
A port with the number specified in the *port_number* parameter has already been defined.

CANT_MODIFY_WHEN_ACTIVE
You have attempted to modify an active port and change parameter values when the port is active. The parameter values you can change while the port is active are:
description
*implicit_**
default_tg_chars
driver_name through *tx_thruput_class*

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

Incoming Calls

If you are configuring a port that accepts incoming calls (as defined by the *tot_link_act_lim*, *inb_link_act_lim*, and *out_link_act_lim* parameters), there is generally no need to define an LS to use for these calls; Communications Server for Linux will dynamically define an LS when the incoming call is received. However, if the incoming calls are from a host computer that supports dependent LUs or from a downstream computer using SNA gateway, you need to explicitly define an LS because the LS definition includes the name of the PU associated with the dependent LUs or the name of the downstream PU.

When an incoming call arrives at the port, Communications Server for Linux checks the address specified on the call against the addresses specified for link stations defined on the port (if any) to determine if an LS has already been defined for the call. If the address does not match, an LS is dynamically defined. To ensure that the explicit LS definition (including the required PU name) is used, be sure that the address defined for this LS matches the address that is supplied by the host or the downstream computer on the incoming call.

define_rcf_access

The **define_rcf_access** command defines access to the Communications Server for Linux Remote Command Facility (RCF). This command defines the user ID used to run UNIX® Command Facility (UCF) commands and the restrictions under which administration commands can be issued using the Service Point Command Facility (SPCF). For more information about SPCF and UCF, refer to the Communications Server for Linux Administration Guide. You can use this command to permit access to SPCF, UCF, or both.

The command can be used to specify the RCF access for the first time, or to modify an existing definition. Because RCF access parameters are defined as domain resources, this command is not associated with a particular node.

Communications Server for Linux acts on these parameters during node start-up; if these parameters are changed while a node is running, the changes do not take effect on the server where the node is running until the node is stopped and restarted.

Supplied Parameters

Parameter name	Type	Length	Default
[define_rcf_access]			
ucf_username	character	31	(null string)
spcf_permissions	constant		NONE

Supplied parameters are:

ucf_username
: Specifies the Linux user name of the UCF user. This parameter is a string of locally displayable characters. Do not specify the name `root`; Communications Server for Linux does not allow UCF commands to be run as root for security reasons.

All UCF commands are run using the user ID for this user, with the default shell, default group ID, and access permissions that are defined on the Linux system for this user.

To prevent access to UCF, do not specify this parameter.

spcf_permissions
Specifies the types of Communications Server for Linux administration commands that can be accessed using SPCF. To prevent access to SPCF, set this parameter to `NONE`. To allow access to SPCF, set this parameter to one or more of the following values (combined using a + character):

ALLOW_QUERY_LOCAL
The **query**_* commands are allowed.

ALLOW_DEFINE_LOCAL
The **define**_*, **set**_*, **delete**_*, **add**_*, **remove**_* , and **init_node** commands are allowed.

ALLOW_ACTION_LOCAL
The **start**_*, **stop**_*, **activate**_*, **deactivate**_*, **aping**, **initialize_session_limit**, **change_session_limit**, and **reset_session_limit** commands are allowed.

ALLOW_QUERY_REMOTE
The **query**_* commands are allowed to be directed at any node in the domain.

ALLOW_DEFINE_REMOTE
The **define**_*, **set**_*, **delete**_*, **add**_*, **remove**_* , and **init_node** commands are allowed to be directed at any node in the domain.

ALLOW_ACTION_REMOTE
The **start**_*, **stop**_*, **activate**_*, **deactivate**_*, **aping**, **initialize_session_limit**, **change_session_limit**, and **reset_session_limit** commands are allowed to be directed at any node in the domain.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
`PARAMETER_CHECK`

secondary_rc
Possible values are:

UCF_USER_CANNOT_BE_ROOT
The *ucf_username* parameter was specified as the name `root` (not allowed).

INVALID_SPCF_SECURITY
The *spcf_permissions* parameter was not set to a valid value.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_rtp_tuning

The **define_rtp_tuning** command specifies parameters to be used when setting up RTP connections. After you issue this command, the parameters you specify will be used for all future RTP connections until you modify them by issuing a new **define_rtp_tuning** command.

Supplied Parameters

Parameter name	Type	Length	Default
[define_rtp_tuning]			
path_switch_attempts	decimal		6
short_req_retry_limit	decimal		6
path_switch_time_low	decimal		480
path_switch_time_medium	decimal		240
path_switch_time_high	decimal		120
path_switch_time_network	decimal		60
refifo_cap	decimal		4000
srt_cap	decimal		8000

Supplied parameters are:

path_switch_attempts
Number of path switch attempts to set on new RTP connections. Specify a value in the range 1–255.

short_req_retry_limit
Number of times a Status Request is sent before Communications Server for Linux determines that an RTP connection is disconnected and starts Path Switch processing. Specify a value in the range 1–255.

path_switch_time_low
Length of time in seconds for which Communications Server for Linux attempts to path switch a disconnected RTP connection with transmission priority set toAP_LOW. Specify a value in the range 1–65535.

path_switch_time_medium
Length of time in seconds for which Communications Server for Linux attempts to path switch a disconnected RTP connection with transmission priority set toAP_MEDIUM. Specify a value in the range 1–65535. The value you specify must not exceed the value of *path_switch_time_low*.

path_switch_time_high
Length of time in seconds for which Communications Server for Linux attempts to path switch a disconnected RTP connection with transmission priority set toAP_HIGH. Specify a value in the range 1–65535. The value you specify must not exceed the value for any lower transmission priority.

path_switch_time_network
Length of time in seconds for which Communications Server for Linux attempts to path switch a disconnected RTP connection with transmission priority set toAP_NETWORK. Specify a value in the range 1–65535. The value you specify must not exceed the value for any lower transmission priority.

refifo_cap
The RTP protocol uses a timer called the Re-FIFO Timer. The value of this timer is calculated as part of the protocol, but this parameter specifies a maximum value in milliseconds beyond which the timer cannot increase. In some situations, setting this maximum value can improve performance.

Specify a value in the range 0–12000. If you specify a value in the range 1–249, the value of 250 will be used. Setting a value of 0 (zero) means that the timer is not limited and can take any value calculated by the protocol.

srt_cap The RTP protocol uses a timer called the Short Request Timer. The value of this timer is calculated as part of the protocol, but this parameter specifies a maximum value in milliseconds beyond which the timer cannot increase. In some situations, setting this maximum value can improve performance.

Specify a value in the range 0–24000. If you specify a value in the range 1–499, the value of 500 will be used. Setting a value of 0 (zero) means that the timer is not limited and can take any value calculated by the protocol.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

INVALID_PATH_SWITCH_TIMES
One or more of the specified path switch times was not valid; for example, you may have specified a value for one transmission priority that exceeds the value specified for a lower transmission priority.

State Check

No specific state errors, other than those listed in Appendix A, “Common Return Codes from snaadmin Commands,” on page 589, occur for this command.

Other Conditions

Appendix A, “Common Return Codes from snaadmin Commands,” on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_sdlc_dlc

The **define_sdlc_dlc** command defines a new SDLC DLC.

The command can also be used to modify an existing DLC, if the DLC is not currently active. However, you cannot change the negotiable link support for an existing DLC; this parameter can be specified only when creating a new DLC.

Supplied Parameters

Parameter name	Type	Length	Default
[define_sdlc_dlc]			
dlc_name	character	8	
description	character	31	(null string)
neg_ls_supp	constant		YES
adapter_number	decimal		0
initially_active	constant		YES

Supplied parameters are:

dlc_name
Name of the DLC. This name is a character string using any locally displayable characters.

description
A text string describing the DLC. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_dlc** command.

neg_ls_supp
Specifies whether the DLC supports negotiable link stations. You cannot change the negotiable link support for an existing DLC; this parameter can be specified only when creating a new DLC. Possible values are as follows:

YES Negotiable link stations are supported; link stations using this DLC can be primary, secondary, or negotiable.

NO Negotiable link stations are not supported; link stations using this DLC must be primary or secondary.

adapter_number
Adapter number used by the DLC. If the server contains more than one SDLC adapter card, specify `0` (zero) for the first card, `1` for the second card, and so on. Otherwise, set this parameter to `0` (zero).

initially_active
Specifies whether this DLC is automatically started when the node is started. Possible values are:

YES The DLC is automatically started when the node is started.

NO The DLC is automatically started only if a port or LS that uses the DLC is defined as initially active; otherwise, it must be manually started.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

INVALID_DLC_NAME
The supplied *dlc_name* parameter contained a character that was not valid.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

DLC_ACTIVE
A parameter cannot be modified because the DLC is currently active.

INVALID_DLC_TYPE
You cannot change the negotiable link support for an existing DLC. This parameter can be specified only when creating a new DLC.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_sdlc_ls

The **define_sdlc_ls** command is used to define a new SDLC link station (LS) or modify an existing one.

Before issuing this command, you must define the port that this LS uses.

You cannot use this command to modify the port used by an existing LS; the name of the port specified in the *port_name* parameter for this command must match the previous definition of the LS. The LS can be modified only if it is not started.

Supplied Parameters

Parameter name	Type	Length	Default
[define_sdlc_ls]			
ls_name	character	8	
description	character	31	(null string)
port_name	character	8	
adj_cp_name	character	17	(null string)
adj_cp_type	constant		LEARN_NODE
address	hex		0xC1
auto_act_supp	constant		NO
tg_number	decimal		0
limited_resource	constant		NO
solicit_sscp_sessions	constant		NO
pu_name	character	8	(taken from ls_name)
disable_remote_act	constant		NO
dspu_services	constant		NONE
dspu_name	character	8	(taken from ls_name)
dlus_name	character	17	(null string)
bkup_dlus_name	character	17	(null string)
hpr_supported	constant		NO
link_deact_timer	decimal		30
default_nn_server	constant		NO
ls_attributes	constant		SNA
adj_node_id	hex array	4	0x00000000
local_node_id	hex array	4	0x00000000
cp_cp_sess_support	constant		YES
use_default_tg_chars	constant		YES
effect_cap	decimal		64000
connect_cost	decimal		0
byte_cost	decimal		0
security	constant		SEC_NONSECURE
prop_delay	constant		PROP_DELAY_TELEPHONE
user_def_parm_1	decimal		128
user_def_parm_2	decimal		128
user_def_parm_3	decimal		128
target_pacing_count	decimal		7
max_send_btu_size	decimal		265
ls_role	constant		USE_PORT_DEFAULTS
conventional_lu_compression	constant		NO
initially_active	constant		NO
react_timer	decimal		30
react_timer_retry	decimal		65535
restart_on_normal_deact	constant		NO
poll_frame	constant		XID
max_ifrm_rcvd	decimal		0
dlus_retry_timeout	decimal		0
dlus_retry_limit	decimal		0
branch_link_type	constant		NONE
dddlu_offline_supported	constant		NO

Supplied parameters are:

ls_name
: Name of the link station to be defined.

description
: A text string describing the LS. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_ls**, **query_pu**, and **query_downstream_pu** commands.

port_name
: Name of the port associated with this link station. This name must match the name of a defined port.

adj_cp_name
: Fully qualified name of the adjacent CP for this LS. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character CP name. This parameter is used in the following ways:

  - If the *adj_cp_type* parameter is set to NETWORK_NODE or END_NODE, and preassigned TG numbers are being used, set this parameter to the CP

name defined at the adjacent node; if the adjacent node sends a CP name during XID exchange, it will be checked against this value.

- If *adj_cp_type* is set to BACK_LEVEL_LEN_NODE, Communications Server for Linux uses this value only as an identifier; set it to any string that does not match other CP names defined at this node.
- If *adj_cp_type* is set to any other value, or if preassigned TG numbers are not being used, there is no need to specify this parameter; Communications Server for Linux checks the CP name only if one is specified.

adj_cp_type
Adjacent node type.

If the adjacent node is an APPN node and preassigned TG numbers are not being used, this parameter is usually set to LEARN_NODE, indicating that the node type is unknown; Communications Server for Linux will determine the type during XID exchange.

If preassigned TG numbers are being used, you must specify the node type explicitly. You can also specify the type as an additional security check if preassigned TG numbers are not being used. In this case, Communications Server for Linux will reject a connection attempt from the adjacent node if its node type does not match the one specified for this parameter. Possible values are:

LEARN_NODE
The adjacent node type is unknown; Communications Server for Linux will determine the type during XID exchange.

END_NODE
The adjacent node is an End Node, a Branch Network Node acting as an End Node from the local node's perspective, or a LEN node that includes the Network Name CV in its XID3.

NETWORK_NODE
The adjacent node is a Network Node, or a Branch Network Node acting as a Network Node from the local node's perspective.

If the adjacent node is not an APPN node, use one of the following values:

BACK_LEVEL_LEN_NODE
The adjacent node is one that does not include the network name control vector (NNCV) in its XID3.

HOST_XID3
The adjacent node is a host node; Communications Server for Linux responds to a polling XID from the node with a format 3 XID.

HOST_XID0
The adjacent node is a host node; Communications Server for Linux responds to a polling XID from the node with a format 0 XID.

DSPU_XID
The adjacent node is a downstream PU; Communications Server for Linux includes XID exchange in link activation. The *dspu_name* and *dspu_services* parameters must also be set.

DSPU_NOXID
The adjacent node is a downstream PU; Communications Server

for Linux does not include XID exchange in link activation. The *dspu_name* and *dspu_services* parameters must also be set.

If you want to run independent LU 6.2 traffic over this LS, you must set the *adj_cp_type* parameter to LEARN_NODE, END_NODE, NETWORK_NODE, or BACK_LEVEL_LEN_NODE.

address Address of the secondary station on this LS.

The value of this parameter depends on how the port that owns this LS is configured, as follows:

- If the *out_link_act_lim*parameter on **define_sdlc_port** is 0 (zero), the port is used only for incoming calls and this parameter is reserved.
- If the port is switched primary and is used for outgoing calls (*port_type* is PORT_SWITCHED, *ls_role* is LS_PRI, and *out_link_act_lim* on **define_sdlc_port** is a nonzero value), either set this parameter to 0xFF to accept whatever address is configured at the secondary station, or set it to a 1-byte value in the range 0x01–0xFE (this value must match the value configured at the secondary station).
- For all other port configurations, set this parameter to a 1-byte value in the range 0x01–0xFE to identify the link station. If the port is primary multi-drop (*ls_role* on **define_sdlc_port** is LS_PRI and *tot_link_act_lim* is greater than 1), this address must be different for each LS on the port.

auto_act_supp
Specifies whether the link can be automatically activated when required by a session. Possible values are:

YES The link can be automatically activated.

The reactivation timer parameters are ignored. If the LS fails, Communications Server for Linux does not attempt to reactivate it until a dependent LU application that was using the session attempts to restart the session; an LS used by independent LUs is not reactivated by Communications Server for Linux and must be manually restarted.

The following restrictions also apply:

- If the link is to an APPN node, the LS must have a preassigned TG number defined (see the *tg_number* parameter), and *cp_cp_sess_support* must be set to NO.
- If either the local node or the adjacent node is an end node, the LS must also be defined to automatically activate at the adjacent node.

NO The link cannot be automatically activated.

tg_number
Preassigned TG number. This parameter is used only if the adjacent node is an APPN node (*adj_cp_type* is either NETWORK_NODE or END_NODE); it is ignored otherwise.

This TG number is used to represent the link when the link is activated. The node does not accept any other number from the adjacent node during activation of this link; if the adjacent node is using preassigned TG numbers, the same TG number must be defined by the adjacent node on the adjacent link station.

If the local node is a LEN node, or if the adjacent node is a LEN node and the link is to be automatically activated, set the TG number to 1.

Otherwise, specify a number in the range 1–20, or 0 (zero) to indicate that the TG number is not preassigned and is negotiated when the link is activated.

If a preassigned TG number is defined, the *adj_cp_name* parameter must also be defined, and the *adj_cp_type* parameter must be set to either END_NODE or NETWORK_NODE.

limited_resource
Specifies whether this link station is a limited resource and is automatically deactivated when there are no sessions using the link. Link stations on a nonswitched port cannot be configured as limited resource. Possible values are:

NO The link is not a limited resource and is not automatically deactivated.

NO_SESSIONS
The link is a limited resource and is automatically deactivated when no active sessions are using it.

INACTIVITY
The link is a limited resource and is automatically deactivated when no active sessions are using it, or when no data has flowed on the link for the time period specified by the *link_deact_timer* parameter.

- If no SSCP-PU session is active across the link, the node deactivates the link immediately.
- If an SSCP-PU session is active but no traffic has flowed for the specified time period, the node sends REQDISCONT(normal) to the host. The host is then responsible for deactivating all LUs and the PU, at which time the local node will deactivate the link. However, the host may not deactivate LUs with active PLU-SLU sessions; in this case, the link remains active until all these sessions are deactivated (for example by the user logging out). This behavior can be changed by using options in the *ptf* parameter of the **start_node** command.

A limited resource link station can be configured for CP-CP session support by setting this parameter to NO_SESSIONS and *cp_cp_sess_support* to YES. In this case, if CP-CP sessions are brought up over the link, Communications Server for Linux does not treat the link as a limited resource and therefore does not deactivate it.

solicit_sscp_sessions
Specifies whether to request the adjacent node to initiate sessions between the SSCP and the local CP and dependent LUs. This parameter is used only if the adjacent node is an APPN node (*adj_cp_type* is either NETWORK_NODE or END_NODE); it is ignored otherwise. If the adjacent node is a host (*adj_cp_type* is either HOST_XID3 or HOST_XID0), Communications Server for Linux always requests the host to initiate SSCP sessions.

Possible values are:

YES Request the adjacent node to initiate SSCP sessions.

NO Do not request the adjacent node to initiate SSCP sessions.

pu_name
Name of the local PU that uses this link. This parameter is required only if

adj_cp_type is set to HOST_XID3 or HOST_XID0, or if *solicit_sscp_sessions* is set to YES; it is ignored otherwise. This name is a type-A character string starting with a letter.

You cannot change the PU name on an LS that is already defined.

If the PU name is required and you do not specify it, the default PU name is the same as the LS name. To ensure that this name is a valid type-A character string, Communications Server for Linux converts it to uppercase; if the string begins with a numeric character, this character is either removed or preceded by the characters "PU."

disable_remote_act
Specifies whether to prevent activation of the LS by the remote node. Possible values are:

YES The LS can be activated only by the local node; if the remote node attempts to activate it, Communications Server for Linux will reject the attempt.

NO The LS can be activated by the remote node.

dspu_services
Specifies the services that the local node provides to the downstream PU across this link. This parameter is used only if the adjacent node is a downstream PU, or an APPN node with *solicit_sscp_sessions* set to NO; it is reserved otherwise. Possible values are:

PU_CONCENTRATION
Local node provides Physical Unit (PU) concentration for the downstream PU. The local node must be defined to support SNA gateway.

DLUR Local node provides DLUR services for the downstream PU. The local node must be defined to support DLUR. (DLUR is not supported on end node.)

NONE Local node provides no services for the downstream PU.

dspu_name
Name of the downstream PU. The name is a type-A character string starting with a letter.

This parameter is reserved except when both of the following conditions are true:

- The *solicit_sscp_sessions* parameter is set to NO
- The *dspu_services* parameter is set to PU_CONCENTRATION or DLUR

If both of these conditions are true and you do not specify a value for *dspu_name*, the default is the same as the LS name. To ensure that this name is a valid type-A character string, Communications Server for Linux converts it to uppercase; if the string begins with a numeric character, this character is either removed or preceded by the characters "PU."

If the downstream PU is used for DLUR, this name should match the PU name configured on the host. (Communications Server for Linux sends both the PU name and PU ID to the host to identify the PU. The host normally identifies the PU by its PU name, or by the PU ID if it cannot find a matching PU name.)

dlus_name
Name of the DLUS node from which DLUR solicits SSCP services when the link to the downstream node is activated. This parameter is reserved if *dspu_services* is not set to DLUR.

The name is a type-A character string, consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character DLUS name.

To specify the global default DLUS defined using the **define_dlur_defaults** command, do not specify this parameter. If this parameter is not specified and there is no global default DLUS, then DLUR will not initiate SSCP contact when the link is activated.

bkup_dlus_name
Name of the backup DLUS node from which DLUR solicits SSCP services if the node specified by *dlus_name* is not active. This parameter is reserved if *dspu_services* is not set to DLUR.

The name is a type-A character string, consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character backup DLUS name.

To specify the global backup default DLUS defined using **define_dlur_defaults**, do not specify this parameter.

hpr_supported
Specifies whether HPR is supported on this link. This parameter must be set to NO unless the *adj_cp_type* parameter indicates that the link connects to an APPN node. Possible values are:

YES HPR is supported on this link.

NO HPR is not supported on this link.

link_deact_timer
Limited resource link deactivation timer, in seconds. A limited resource link is automatically deactivated if no data flows over the link for the time specified by this parameter. This parameter is not used if *limited_resource* is set to any value other than INACTIVITY.

The minimum value is 5; values in the range 1–4 will be interpreted as 5.

The value 0 (zero) indicates one of the following:

- If the *hpr_supported* parameter is set to YES, the default deactivation timer value of 30 is used.
- If the *hpr_supported* parameter is set to NO, no timeout is used (the link is not deactivated, as if *limited_resource* were set to NO).

default_nn_server
For an end node this parameter specifies whether the link station being defined supports CP-CP sessions to a network node that can act as the local node's network node server. When the local node has no CP-CP sessions to a network node server and needs to establish them, the local node checks this parameter on its defined link stations to find a suitable LS to activate. This enables you to specify which adjacent NNs are suitable to act as the NN server (for example, to avoid using NNs that are accessed by expensive or slow links).

Possible values are:

YES This link supports CP-CP sessions to a network node that can act

as the local node's NN server; the local node can automatically activate this link if it needs to contact an NN server. The *cp_cp_sess_support* parameter must be set to YES.

NO This link does not support CP-CP sessions to a network node that can act as the local node's NN server; the local node cannot automatically activate this link if it needs to contact an NN server.

If the local node is not an end node, this parameter is ignored.

ls_attributes

Attributes of the remote system with which Communications Server for Linux is communicating.

Specify SNA unless you are communicating with a host of one of the other following types. Possible values are:

SNA Standard SNA host

FNA Fujitsu Network Architecture (VTAM-F) host

HNA Hitachi Network Architecture host

SUPPRESS_CP_NAME
Suppress the CP name associated with the remote node. Use a + character to combine this value with SNA, FNA, or HNA.

If *adj_cp_type* is set to BACK_LEVEL_LEN_NODE, and the remote LEN node associated with this LS cannot accept the Network Name CV in the format 3 XID it receives, use a + character to combine the value SNA, FNA, or HNA with SUPPRESS_CP_NAME (for example, SNA+SUPPRESS_CP_NAME).

If *adj_cp_type* is set to any other value, the SUPPRESS_CP_NAME option is ignored.

adj_node_id

Node ID of adjacent node. This ID is a 4-byte hexadecimal string, consisting of a block number (three hexadecimal digits) and a node number (five hexadecimal digits). To disable node ID checking, do not specify this parameter. If this link station is defined on a switched port, the *adj_node_id* must be unique, and there may only be one null *adj_node_id* on each switched port.

local_node_id

Node ID sent in XIDs on this LS. This ID is a 4-byte hexadecimal string, consisting of a block number (three hexadecimal digits) and a node number (five hexadecimal digits). To use the node ID specified in the *node_id* parameter on the **define_node** command, do not specify this parameter.

cp_cp_sess_support

Specifies whether CP-CP sessions are supported. This parameter is valid only if the adjacent node is an end node or a network node (*adj_cp_type* is NETWORK_NODE, END_NODE, or LEARN_NODE); it is ignored otherwise. If both the local node and the adjacent node are network nodes, this parameter should be set to YES in order to use APPN functions between these nodes.

Possible values are:

YES CP-CP sessions are supported.

NO CP-CP sessions are not supported.

use_default_tg_chars
Specifies whether to use the default TG characteristics supplied on **define_sdlc_port**. The TG characteristics apply only if the link is to an APPN node. If the link is not to an APPN node, the *use_default_tg_chars* through *user_def_parm_3* parameters are ignored. Possible values are:

YES Use the default TG characteristics; ignore the *effect_cap* through *user_def_parm_3* parameters on this command.

NO Use the *effect_cap* through *user_def_parm_3* parameters on this command.

effect_cap
A decimal value representing the line speed in bits per second.

connect_cost
Cost per connect time. Valid values are integer values in the range `0`–`255`, where `0` is the lowest cost per connect time and `255` is the highest cost per connect time.

byte_cost
Cost per byte. Valid values are integer values in the range `0`–`255`, where `0` is the lowest cost per byte and `255` is the highest cost per byte.

security
Security level of the network. Possible values are:

SEC_NONSECURE
No security.

SEC_PUBLIC_SWITCHED_NETWORK
Data is transmitted over a public switched network.

SEC_UNDERGROUND_CABLE
Data is transmitted over secure underground cable.

SEC_SECURE_CONDUIT
Data is transmitted over a line in a secure conduit that is not guarded.

SEC_GUARDED_CONDUIT
Data is transmitted over a line in a conduit that is protected against physical tapping.

SEC_ENCRYPTED
Data is encrypted before transmission over the line.

SEC_GUARDED_RADIATION
Data is transmitted over a line that is protected against physical and radiation tapping.

prop_delay
Propagation delay (the time that a signal takes to travel the length of the link). Specify one of the following values, according to the type of link:

PROP_DELAY_MINIMUM
Minimum propagation delay.

PROP_DELAY_LAN
Delay is less than .5 microseconds (typical for a LAN).

PROP_DELAY_TELEPHONE
Delay is in the range .5–50 microseconds (typical for a telephone network).

PROP_DELAY_PKT_SWITCHED_NET
Delay is in the range 50–250 microseconds (typical for a packet-switched network).

PROP_DELAY_SATELLITE
Delay is greater than 250 microseconds (typical for a satellite link).

PROP_DELAY_MAXIMUM
Maximum propagation delay.

user_def_parm_1 **through** *user_def_parm_3*
User-defined parameters, that you can use to include other TG characteristics not covered by the previous parameters. Each of these parameters must be set to a value in the range 0–255.

target_pacing_count
Indicates the desired pacing window size. Specify a value in the range 1–32,767.

max_send_btu_size
Maximum BTU size that can be sent from this link station. This value is used to negotiate the maximum BTU size that a pair of link stations can use to communicate with each other. The value includes the length of the TH and RH (total 9 bytes) as well as the RU. Specify a value in the range 265–4105.

ls_role Link station role. This parameter is usually set to USE_PORT_DEFAULTS, specifying that the LS role is to be taken from the definition of the port that owns this LS.

If you need to override the port's LS role for an individual LS, specify one of the following values:

LS_PRI Primary

LS_SEC Secondary

LS_NEG Negotiable

conventional_lu_compression
Specifies whether data compression is requested for LU 0–3 sessions on this link. This parameter is used only if this link carries LU 0–3 traffic; it does not apply to LU 6.2 sessions. Possible values are:

YES Data compression should be used for LU 0–3 sessions on this link if the host requests it.

NO Data compression should not be used for LU 0–3 sessions on this link.

initially_active
Specifies whether this LS is automatically started when the node is started. Possible values are:

YES The LS is automatically started when the node is started.

NO The LS is not automatically started; it must be manually started.

If the LS is a leased link, you are recommended to set this parameter to YES to ensure that the link is always available.

react_timer
Reactivation timer for reactivating a failed LS. If the *react_timer_retry* parameter is a nonzero value (to specify that Communications Server for Linux should retry activating the LS if it fails), this parameter specifies the

time in seconds between retries. When the LS fails, or when an attempt to reactivate it fails, Communications Server for Linux waits for the specified time before retrying the activation. If *react_timer_retry* is `0` (zero), this parameter is ignored.

react_timer_retry
Retry count for reactivating a failed LS. This parameter is used to specify whether Communications Server for Linux should attempt to reactivate the LS if it fails while in use (or if an attempt to start the LS fails).

Specify `0` (zero) to indicate that Communications Server for Linux should not attempt to reactivate the LS, or specify the number of retries to be made. A value of `65,535` indicates that Communications Server for Linux should retry indefinitely until the LS is reactivated.

Communications Server for Linux waits for the time specified by the *react_timer* parameter between successive retries. If the LS is not successfully reactivated by the end of the retry count, or if **stop_ls** is issued while Communications Server for Linux is retrying the activation, no further retries are made; the LS remains inactive unless **start_ls** is issued for it.

If the *auto_act_supp* parameter is set to `YES`, the *react_timer* and *react_timer_retry* parameters are ignored; if the link fails, Communications Server for Linux does not attempt to reactivate it until the user application that was using the session attempts to restart the session.

If the LS is a leased SDLC link, you are recommended to set this parameter to a non-zero value to ensure that the link is always available.

restart_on_normal_deact
Specifies whether Communications Server for Linux should attempt to reactivate the LS if it is deactivated normally by the remote system. Possible values are:

YES If the remote system deactivates the LS normally, Communications Server for Linux attempts to reactivate it, using the same retry timer and count values as for reactivating a failed LS (the *react_timer* and *react_timer_retry* parameters above).

NO If the remote system deactivates the LS normally, Communications Server for Linux does not attempt to reactivate it.

If the LS is a host link (specified by the *adj_cp_type* parameter), or is automatically started when the node is started (the *initially_active* parameter is set to `YES`), this parameter is ignored; Communications Server for Linux always attempts to reactivate the LS if it is deactivated normally by the remote system (unless *react_timer_retry* is zero).

poll_frame
The frame to use for preactivation polling. This frame is usually `XID`, indicating that polling is in the control of the DLC user. However, when Communications Server for Linux is primary talking to an old secondary implementation, it may be necessary to poll using some other frame. Possible values are:

```
XID
SNRM
```

max_ifrm_rcvd
Maximum number of I-frames that can be received by the local link stations before an acknowledgment is sent. Specify a value in the range 1–127.

dlus_retry_timeout
The interval in seconds between the second and subsequent attempts to contact the DLUS specified by the *dlus_name* and *bkup_dlus_name* parameters. The interval between the first and second attempts is always 1 second.

Specify a value in the range 0– 65,535. If you specify 0, the default specified using **define_dlur_defaults** is used. This parameter is ignored if the *dspu_services* parameter is not set to DLUR.

dlus_retry_limit
Retry count for contacting a DLUS. This parameter is used to specify the number of times Communications Server for Linux should retry if it fails to contact the DLUS on the first attempt.

Specify a value in the range 1–65,534, or specify 65,535 to indicate that Communications Server for Linux should retry indefinitely until it contacts the DLUS.

branch_link_type
This parameter applies only if the local node is a Branch Network Node; it is not used if the local node is any other type.

If the parameter *adj_cp_type* is set to NETWORK_NODE, END_NODE, APPN_NODE, or BACK_LEVEL_LEN_NODE, this parameter defines whether the link is an uplink or a downlink. Possible values are:

UPLINK The link is an uplink.

DOWNLINK
The link is a downlink.

If *adj_cp_type* is set to NETWORK_NODE, this parameter must be set to UPLINK.

adj_brnn_cp_support
This parameter applies only if the local node is a Branch Network Node and the adjacent node is a network node (the parameter *adj_cp_type* is set to NETWORK_NODE, or it is set to APPN_NODE and the node type discovered during XID exchange is network node). It is not used if the local and remote nodes are any other type.

This parameter defines whether the adjacent node can be a Branch Network Node that is acting as a Network Node from the point of view of the local node. Possible values are:

ALLOWED
The adjacent node is allowed (but not required) to be a Branch Network Node.

REQUIRED
The adjacent node must be a Branch Network Node.

PROHIBITED
The adjacent node must not be a Branch Network Node.

If *adj_cp_type* is set to NETWORK_NODE and *auto_act_supp* is set to YES, this parameter must be set to REQUIRED or PROHIBITED.

dddlu_offline_supported
Specifies whether the local PU should send NMVT (power off) messages to the host. If the host system supports DDDLU (Dynamic Definition of Dependent LUs), Communications Server for Linux sends NMVT (power off) to the host when it has finished using a dynamically defined LU. This allows the host to save resources by removing the definition when it is no longer required.

This parameter is used only if this link is to a host (*solicit_sscp_sessions* is set to YES and *dspu_services* is not set to NONE).

Possible values are:

YES The local PU sends NMVT (power off) messages to the host.

NO The local PU does not send NMVT (power off) messages to the host.

If the host supports DDDLU but does not support the NMVT (power off) message, this parameter must be set to NO.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

CANT_MODIFY_PORT_NAME
The *ls_name* parameter matched the name of an existing LS, but the *port_name* parameter did not match the existing definition. You cannot modify the port name when changing the definition of an existing LS.

DEF_LINK_INVALID_SECURITY
The *security* parameter was not set to a valid value.

INVALID_AUTO_ACT_SUPP
The *auto_act_supp* parameter was not set to a valid value or was set to YES when *cp_cp_sess_support* was also set to YES.

INVALID_CP_NAME
The *adj_cp_name* parameter contained a character that was either not valid, not in the correct format, or not specified when required.

INVALID_LIMITED_RESOURCE
The *limited_resource* parameter was not set to a valid value.

INVALID_LINK_NAME
The *ls_name* parameter contained a character that was not valid.

INVALID_LS_ROLE
The *ls_role* parameter was not set to a valid value.

INVALID_NODE_TYPE
The *adj_cp_type* parameter was not set to a valid value.

INVALID_PORT_NAME
The *port_name* parameter did not match the name of any defined port.

INVALID_PU_NAME
The *pu_name* parameter did not match the name of any defined PU, or was set to a new value on an already-defined LS.

INVALID_DSPU_NAME
The *dspu_name* parameter did not match the name of any defined PU or was set when not expected.

INVALID_DSPU_SERVICES
The *dspu_services* parameter was not set to a valid value, or was set to a new value on an already-defined LS.

INVALID_SOLICIT_SSCP_SESS
The *solicit_sscp_sess* parameter was not set to a valid value.

INVALID_TARGET_PACING_CNT
The *target_pacing_count* parameter was not set to a valid value.

INVALID_DLUS_NAME
The *dlus_name* parameter contained a character that was not valid or was not in the correct format.

INVALID_BKUP_DLUS_NAME
The *bkup_dlus_name* parameter contained a character that was not valid or was not in the correct format.

HPR_NOT_SUPPORTED
A reserved parameter was set to a nonzero value.

INVALID_TG_NUMBER
The TG number supplied was not in the valid range.

MISSING_CP_NAME
A TG number was defined, but no CP name was supplied.

MISSING_CP_TYPE
A TG number was defined, but no CP type was supplied.

MISSING_TG_NUMBER
The link was defined as automatically activated, but no TG number was supplied.

INVALID_BRANCH_LINK_TYPE
The *branch_link_type* parameter was not set to a valid value.

INVALID_BRNN_SUPPORT
The *adj_brnn_cp_support* parameter was not set to a valid value.

BRNN_SUPPORT_MISSING
The *adj_brnn_cp_support* parameter was set to `ALLOWED`; this value is not valid because the adjacent node is a Network Node and *auto_act_supp* is set to `YES`.

INVALID_UPLINK
The *branch_link_type* parameter was set to `UPLINK`, but the definition

of an existing LS between the local and adjacent nodes specifies that it is a downlink. The branch link type must be the same for all LSs between the same two nodes.

INVALID_DOWNLINK
The *branch_link_type* parameter was set to DOWNLINK, but the definition of an existing LS between the local and adjacent nodes specifies that it is an uplink. The branch link type must be the same for all LSs between the same two nodes.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

DUPLICATE_CP_NAME
The CP name specified in the *adj_cp_name* parameter has already been defined.

DUPLICATE_DEST_ADDR
The destination address specified in the *address* parameter has already been defined.

INVALID_LINK_NAME
The link station value specified in the *ls_name* parameter was not valid.

INVALID_NUM_LS_SPECIFIED
The number of link stations specified was not valid.

LOCAL_CP_NAME
The value specified in the *adj_cp_name* parameter was the same as the local CP name.

LS_ACTIVE
The link station specified in the *ls_name* parameter is currently active.

PU_ALREADY_DEFINED
The PU specified in the *pu_name* parameter has already been defined.

DSPU_ALREADY_DEFINED
The downstream PU specified in the *dspu_name* parameter has already been defined.

DSPU_SERVICES_NOT_SUPPORTED
PU_CONCENTRATION or DLUR has been specified on the *dspu_services* parameter, but the node does not support it.

DUPLICATE_TG_NUMBER
The TG number specified in the *tg_number* parameter has already been defined.

TG_NUMBER_IN_USE
The TG number specified in the *tg_number* parameter is in use by another link station.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

Modem Control Characters

If you need to include non-printable control characters in the *hmod_data* parameter, you can use any of the following methods:

- Include one or more of the escape sequences listed in Table 2, preceded and followed by / (forward slash) characters. For example, to include the CR (carriage-return) character, include /CR/.
- Include the decimal value of the control character, preceded and followed by / (forward slash) characters. For example, to include the control character with value 135, include /135/.
- Specify the parameter as a hexadecimal array instead of a character string, so that each character in the string is specified by a pair of hexadecimal digits instead of a printable character or escape sequence.

Table 2. Escape Sequences for Modem Control Characters

Escape Sequence	Decimal Value	Hexadecimal Value
NUL	0	0x00
SOH	1	0x01
STX	2	0x02
ETX	3	0x03
EOT	4	0x04
ENQ	5	0x05
ACK	6	0x06
BEL	7	0x07
BS	8	0x08
HT	9	0x09
LF	10	0x0A
VT	11	0x0B
FF	12	0x0C
CR	13	0x0D
SO	14	0x0E
SI	15	0x0F
DLE	16	0x10
DC1	17	0x11
DC2	18	0x12
DC3	19	0x13
DC4	20	0x14
NAK	21	0x15
SYN	22	0x16
ETB	23	0x17
CAN	24	0x18
EM	25	0x19
SUB	26	0x1A
ESC	27	0x1B
FS	28	0x1C
GS	29	0x1D
RS	30	0x1E
US	31	0x1F
SP	32	0x20

Table 2. Escape Sequences for Modem Control Characters (continued)

Escape Sequence	Decimal Value	Hexadecimal Value
DEL	127	0x7F

define_sdlc_port

The **define_sdlc_port** command is used to define a new SDLC port or to modify an existing port.

Before issuing this command, you must define the DLC that this port uses.

You can modify an existing port only if it is not started. You cannot change the DLC used by an existing port; the *dlc_name* specified when modifying an existing port must match the DLC that was specified on the initial definition of the port.

For information about defining a port that will accept incoming calls, see "Incoming Calls" on page 183.

Supplied Parameters

```
Parameter name                Type          Length    Default
[define_sdlc_port]
port_name                     character     8
description                   character     31        (null string)
dlc_name                      character     8
port_type                     constant                PORT_NON_SWITCHED
port_number                   decimal                 0
max_rcv_btu_size              decimal                 265
tot_link_act_lim              decimal                 1
inb_link_act_lim              decimal                 0
out_link_act_lim              decimal                 1
ls_role                       constant                LS_SEC
act_xid_exchange_limit        decimal                 10
nonact_xid_exchange_limit     decimal                 10
ls_xmit_rcv_cap               constant                LS_TWA
max_ifrm_rcvd                 decimal                 7
target_pacing_count           decimal                 7
max_send_btu_size             decimal                 265
address                       hex number              0x00
implicit_cp_cp_sess_support   constant                YES
implicit_limited_resource     constant                NO
implicit_hpr_support          constant                NO
implicit_deact_timer          decimal                 30
implicit_uplink_to_en         constant                NO
effect_cap                    decimal                 64000
connect_cost                  decimal                 0
byte_cost                     decimal                 0
security                      constant                SEC_NONSECURE
prop_delay                    constant                PROP_DELAY_TELEPHONE
user_def_parm_1               decimal                 128
user_def_parm_2               decimal                 128
user_def_parm_3               decimal                 128
initially_active              constant                YES
implicit_dspu_template        character     8         (null string)
implicit_dspu_services        constant                NONE
implicit_ls_limit             decimal                 0
```

Supplied parameters are:

port_name
Name of the port to be defined. This name is a character string using any locally displayable characters.

description
A text string describing the port. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_port** command.

dlc_name
Name of the associated DLC. This name is a character string using any locally displayable characters. The specified DLC must have already been defined.

port_type
Type of line used by the port. Possible values are:

PORT_SWITCHED
Switched line.

PORT_NONSWITCHED
Non-switched line.

port_number
The number of the port.

max_rcv_btu_size
Maximum BTU size that can be received. This value includes the length of the TH and RH (total 9 bytes), as well as the RU. Specify a value in the range 265–4105.

tot_link_act_lim
Total link activation limit (the maximum number of links that can be active at any time using this port).

If *port_type* is set to PORT_NONSWITCHED and *ls_role* is set to LS_PRI or LS_SEC, the range is 1–254. If a value greater than 1 is specified, the port is defined as multi-drop (primary) or multi-PU (secondary). In all other cases, this parameter must be set to 1.

inb_link_act_lim
Inbound link activation limit (the number of links reserved for inbound activation). The sum of *inb_link_act_lim* and *out_link_act_lim* must not exceed *tot_link_act_lim*; the difference between *inb_link_act_lim* and *tot_link_act_lim* defines the maximum number of links that can be activated outbound at any time.

If *port_type* is set to PORT_NONSWITCHED, this parameter must be 0 (zero). If *port_type* is set to PORT_SWITCHED, then the port must be defined to accept either incoming calls (by setting *inb_link_act_lim* = 1 and *out_link_act_lim* = 0) or outgoing calls (by setting *inb_link_act_lim* = 0 and *out_link_act_lim* = 1).

out_link_act_lim
Outbound link activation limit (the number of links reserved for outbound activation). The sum of *inb_link_act_lim* and *out_link_act_lim* must not exceed *tot_link_act_lim*; the difference between *out_link_act_lim* and *tot_link_act_lim* defines the maximum number of links that can be activated inbound at any time.

If *port_type* is set to PORT_NONSWITCHED, this parameter must be equal to *tot_link_act_lim*. If *port_type* is set to PORT_SWITCHED, then the port must be

defined to accept either incoming calls (by setting *inb_link_act_lim* = 1 and *out_link_act_lim* = 0) or outgoing calls (by setting *inb_link_act_lim* = 0 and *out_link_act_lim* = 1).

ls_role Link station role. Possible values are:

LS_PRI Primary

LS_SEC Secondary

LS_NEG Negotiable

act_xid_exchange_limit
Activation XID exchange limit. Specify a value in the range 1–65,535.

nonact_xid_exchange_limit
Nonactivation XID exchange limit. Specify a value in the range 1–65,535.

ls_xmit_rcv_cap
Specifies the link station transmit/receive capability. Possible values are:

LS_TWS Two-way simultaneous

LS_TWA Two-way alternating

max_ifrm_rcvd
Maximum number of I-frames that can be received by the local link stations before an acknowledgment is sent. Specify a value in the range 1–127.

target_pacing_count
Indicates the desired pacing window size. Specify a value in the range 1–32,767.

max_send_btu_size
Maximum BTU size that can be sent from this port. This value is used to negotiate the maximum BTU size that a pair of link stations can use to communicate with each other. The value includes the length of the TH and RH (total 9 bytes), as well as the RU. Specify a value in the range 265–4105.

address Port address used for incoming calls.

The value of this parameter depends on how the port that owns this LS is configured, as follows:

- If the port is used only for incoming calls, or if *ls_role* is set to LS_PRI, or if *ls_role* is set to LS_NEG and the local station becomes primary after LS role negotiation, this parameter is reserved.
- If *ls_role* is set to LS_SEC, or if *ls_role* is set to LS_NEG and the local station becomes secondary after LS role negotiation, this address is used in the response to an incoming call.

If the address of the remote station is unknown, set this parameter to zero.

implicit_cp_cp_sess_support
Specifies whether CP-CP sessions are allowed for implicit link stations using this port. Possible values are:

YES CP-CP sessions are allowed for implicit link stations.

NO CP-CP sessions are not allowed for implicit link stations.

implicit_limited_resource
Specifies whether implicit link stations off this port are defined as limited resources. Possible values are:

NO Implicit links are not limited resources and are not automatically deactivated.

NO_SESSIONS
Implicit links are limited resources and are automatically deactivated when no active sessions are using them.

INACTIVITY
Implicit links are limited resources and are automatically deactivated when no active sessions are using them or when no data has flowed for the time period specified by the *implicit_deact_timer* parameter.

- If no SSCP-PU session is active across the link, the node deactivates the link immediately.
- If an SSCP-PU session is active but no traffic has flowed for the specified time period, the node sends REQDISCONT(normal) to the host. The host is then responsible for deactivating all LUs and the PU, at which time the local node will deactivate the link. However, the host may not deactivate LUs with active PLU-SLU sessions; in this case, the link remains active until all these sessions are deactivated (for example by the user logging out). This behavior can be changed by using options in the *ptf* parameter of the **start_node** command.

implicit_hpr_support
Specifies whether High Performance Routing (HPR) is supported on implicit links. Possible values are:

YES HPR is supported on implicit links.

NO HPR is not supported on implicit links.

implicit_deact_timer
Implicit limited resource link deactivation timer, in seconds.

If *implicit_hpr_support* is set to `YES` and *implicit_limited_resource* is set to `NO_SESSIONS`, then the implicit link is automatically deactivated if no data flows on it for the time specified by this parameter and no sessions are using the link.

If *implicit_limited_resource* is set to `INACTIVITY`, an implicit link using this port is automatically deactivated if no data flows on it for the time specified by this parameter.

The minimum value is `5`; values in the range `1`–`4` will be interpreted as `5`. The value `0` (zero) indicates no timeout (the link is not deactivated, as though *implicit_limited_resource* were set to `NO`). This parameter is reserved if *implicit_limited_resource* is set to `NO`.

implicit_uplink_to_en
This parameter applies only if the local node is a Branch Network Node; it is reserved if the local node is any other type.

If the adjacent node is an end node, this parameter specifies whether implicit link stations off this port are uplink or downlink. This parameter is ignored if there are existing links to the same adjacent node, because in this case the existing links are used to determine the link type. Possible values are:

YES Implicit links to an End Node are uplinks.

NO Implicit links to an End Node are downlinks.

effect_cap **through** *user_def_parm_3*
Default TG characteristics used for implicit link stations using this port and as the default TG characteristics for defined link stations that do not have explicitly defined TG characteristics. The TG characteristics parameters are ignored if the LS is to a downstream PU.

For more information about these parameters, see "define_tr_ls, define_ethernet_ls" on page 209.

initially_active
Specifies whether this port is automatically started when the node is started. Possible values are:

YES The port is automatically started when the node is started.

NO The port is automatically started only if an LS that uses the port is defined as initially active; otherwise, it must be manually started.

implicit_dspu_template
Specifies the DSPU template, defined on the **define_dspu_template** command. This template is used for definitions if the local node is to provide SNA gateway for an implicit link activated on this port. If the template specified does not exist or is already at its instance limit when the link is activated, activation will fail. This template name is an 8-byte string in a locally displayable character set.

If the *implicit_dspu_services* parameter is not set to `PU_CONCENTRATION`, the *implicit_dspu_template* parameter is reserved.

implicit_dspu_services
Specifies the services that the local node will provide to the downstream PU across implicit links activated on this port. Possible values are:

DLUR Local node will provide DLUR services for the downstream PU (using the default DLUS configured through the **define_dlur_defaults** command).

PU_CONCENTRATION
Local node will provide SNA gateway for the downstream PU. It will also put in place definitions as specified by the DSPU template specified for the parameter *implicit_dspu_template*.

NONE Local node will provide no services for this downstream PU.

implicit_ls_limit
Specifies the maximum number of implicit link stations which can be active on this port simultaneously, including dynamic links and links activated for Discovery. Specify a value in the range `1`–`65,534` or specify `0` (zero) to indicate no limit. A value of `NO_IMPLICIT_LINKS` indicates that no implicit links are allowed.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_PORT_NAME
The value specified in the *port_name* parameter was not valid.

INVALID_DLC_NAME
The specified *dlc_name* did not match any defined DLC.

INVALID_PORT_TYPE
The *port_type* parameter was not set to a valid value.

INVALID_BTU_SIZE
The *max_rcv_btu_size* parameter was not set to a valid value.

INVALID_LS_ROLE
The *ls_role* parameter was not set to a valid value.

INVALID_LINK_ACTIVE_LIMIT
One of the activation limit parameters, *inb_link_act_lim*, *out_link_act_lim*, or *tot_link_act_lim*, was not set to a valid value.

INVALID_MAX_IFRM_RCVD
The *max_ifrm_rcvd* parameter was not set to a valid value.

INVALID_HPR_SUPPORTED
The *implicit_hpr_support* parameter was not set to a valid value.

INVALID_IMPLICIT_UPLINK
The *implicit_uplink_to_en* parameter was not set to a valid value.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

PORT_ACTIVE
The specified port cannot be modified because it is currently active.

DUPLICATE_PORT_NUMBER
A port with the number specified in the *port_number* parameter has already been defined.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

Incoming Calls

If you are configuring a port that accepts incoming calls (as defined by the *tot_link_act_lim*, *inb_link_act_lim*, and *out_link_act_lim* parameters), there is generally

no need to define an LS to use for these calls; Communications Server for Linux will dynamically define an LS when the incoming call is received. However, if the incoming calls are from a host computer that supports dependent LUsor from a downstream computer using SNA gateway, you need to explicitly define an LS because the LS definition includes the name of the PU associated with the dependent LUs or the name of the downstream PU.

When an incoming call arrives at the port, Communications Server for Linux checks the address specified on the call against the addresses specified for link stations defined on the port (if any) to determine if an LS has already been defined for the call. If the address does not match, an LS is dynamically defined. To ensure that the explicit LS definition (including the required PU name) is used, be sure that the address defined for this LS matches the address that is supplied by the host or the downstream computer on the incoming call.

define_security_access_list

The **define_security_access_list** command defines a list of users who can access a particular local LU or invokable TP, so that access to that LU or TP is restricted to the named users. It can also be used to add user names to an existing security access list. The user names in the list are defined using the **define_userid_password** command.

To restrict access for a particular local LU or invokable TP, you need to do the following.

1. Ensure that each authorized user of the LU or TP is defined using the **define_userid_password** command.
2. Use the **define_security_access_list** command to define a security access list containing all of these user IDs.
3. Specify the name of this security access list on the **define_local_lu** or **define_tp** command that defines the LU or TP.

When an incoming Allocate request arrives for a local LU or an invokable TP that has a security access list defined, the invoking application must indicate that conversation security is to be used, and specify a user ID. In addition to the standard conversation security checking (against user IDs specified using the **define_userid_password** command), Communications Server for Linux checks the user ID in the incoming allocate request against the security access list defined for the LU or TP, and rejects the conversation if the user ID does not match. If both the LU and the TP have security access lists defined, the user ID must be in both lists.

If a local LU or an invokable TP does not have a security access list defined, but is still configured to require conversation security, the standard conversation security checking still applies.

Supplied Parameters

Parameter name	Type	Length	Default
[define_security_access_list]			
list_name	character	14	
description	character	31	(null string)
{security_user_data}			
user_name	character	10	

Supplied parameters are:

list_name
The name of the security access list. This name is a character string of 1–14 locally displayable characters.

If this name matches an existing security access list, the users defined by this command are added to the list; otherwise a new list is created.

description
An optional string of 0–31 characters. Communications Server for Linux uses this string for information only. It is stored in the configuration file and returned on the **query_security_access_list** command.

One or more `security_user_data` subrecords may follow. Each subrecord contains the following additional parameter:

user_name
Name of the user.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
`PARAMETER_CHECK`

secondary_rc
One of the following:

INVALID_LIST_NAME
The supplied *list_name* parameter contained a character that was not valid.

INVALID_USER_NAME
One or more of the specified user names was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_tn3270_access

The **define_tn3270_access** command identifies which TN3270 clients, based on their IP addresses, can use the TN3270 server feature of Communications Server for Linux to access a host for 3270 emulation and defines the 3270 LU sessions available to that user. (To define access details for a client using TN Redirector, use **define_tn_redirect**.)

This command can be used to define a new client, to define new sessions for use by an existing client, or to modify the session parameters for an existing client. (To delete sessions from an existing client, use **delete_tn3270_access**.)

Supplied Parameters

Parameter name	Type	Length	Default
[define_tn3270_access]			
default_record	constant		NO
client_address	character	255	(null string)
description	character	31	(null string)
address_format	constant		(IP_ADDRESS)
{tn3270_session_data}			
description	character	31	(null string)
tn3270_support	constant		TN3270E
allow_specific_lu	constant		YES
printer_lu_name	character	8	(null string)
port_number	decimal		
listen_local_address	character	45	(null string)
lu_name	character	8	(null string)
ssl_enabled	constant		NO
security_level	constant		SSL_AUTHENTICATE_MIN
cert_key_label	character	80	(null string)
allow_ssl_timeout_to_nonssl	constant		NO

(One or more tn3270_session_data subrecords can be included.)

Supplied parameters are:

default_record
: Specifies whether **define_tn3270_access** defines the default access record. The default access record is used by a client whose TCP/IP address does not match an address defined by a previous **define_tn3270_access** command. Possible values are:

  YES This command defines the default access record. Do not specify the *client_address* and *address_format* parameters.

  NO This command defines the access record for a specific client.

  A default record provides any client with access to the TN server function (regardless of client address). To restrict TN server use to specific clients, either do not define a default record, or define a default record with no access by not specifying values for the *lu_name* or *printer_lu_name* parameters and setting the *allow_specific_lu* parameter to NO (these parameters are contained in the tn3270_session_data subrecord).

client_address
: The TCP/IP address of the computer on which the TN3270 emulator runs. This can be specified as any of the following; the *address_format* parameter indicates whether it is an IP address or a name.

  - An IPv4 dotted-decimal address (such as 193.1.11.100).

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

If you use a name or alias, the following restrictions apply:

- TN server 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; do not use names that will expand to the same fully qualified name.
- Fully-qualified names are not case-sensitive; for example, `Newbox.THIS.CO.UK` is equivalent to `newbox.this.co.uk`.

description
An optional string of 0–31 characters. You can use it to store additional information to help distinguish between clients. Communications Server for Linux uses this string for information only. It is stored in the configuration file and returned on the **query_tn3270_access_def** command.

address_format
Specifies the format of the *client_address* parameter. Possible values are:

IP_ADDRESS
IP address (either IPv4 or IPv6)

FULLY_QUALIFIED_NAME
Alias or fully qualified name

The following subrecord contains additional parameters:

tn3270_session_data
Each client can access the same TN server node with multiple sessions. For each of these sessions, include a `tn3270_session_data` subrecord with the following additional parameters:

description
An optional string of 0–31 characters. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on a **query_tn3270_access_def** command.

tn3270_support
Specifies the level of TN3270 support. Possible values are:

TN3270 Specifies that TN3270E protocols are disabled.

TN3270E
Specifies that TN3270E protocols are enabled.

TN3270 and TN3287 protocols are always enabled.

To connect an AS/400® TN3270 client, set this parameter to `TN3270E`.

allow_specific_lu
Indicates whether access to specific LUs is allowed. Possible values are:

YES Clients can request access to specific LUs or LU pools instead of using the LU defined by the *lu_name* parameter or *printer_lu_name* parameter on this command.

NO Clients are not allowed to request access to specific LUs.

printer_lu_name
Name of the printer LU or LU pool that this session uses for connections requesting a generic printer LU. This name is an 8-byte type-A character string. The printer LU name must match the name of an LU type 0–3 printer LU defined on this node, or an LU pool containing printer LUs on this node.

If a single printer LU is specified, this printer LU should not be associated with any display LU by the **define_tn3270_association** command. If a printer LU pool is specified, none of the printer LUs in the pool should be associated with display LUs. Allowing a single LU to be accessed as both a generic printer LU and as an associated printer LU may result in the LU not being available as an associated printer LU because it is already in use.

This parameter has no effect if the client requests access to a specific printer LU.

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

If two or more session subrecords use the same *port_number* (for the same *client_address* or a different one), the *listen_local_address* parameter must be specified on all of them or none of them; you cannot specify it on some sessions but leave it unspecified on others.

listen_local_address
The address on the local TN Server computer to which TN3270 clients will connect. This parameter is optional.

- If TN3270 clients are to be able to connect on any local address, or if there is only one valid local address on the TN Server, do not specify this parameter. In this case, any `tn3270_session_data` subrecord that uses the same *port_number* as this one (for the same *client_address* or a different one) must also leave this parameter unspecified.
- If you need to restrict TN3270 clients to a particular local address, specify it as any of the following:
  - An IPv4 dotted-decimal address (such as `193.1.11.100`).
  - An IPv6 colon-hexadecimal address (such as `2001:0db8:0000:0000:0000:0000:1428:57ab` or `2001:db8::1428:57ab`).

  In this case, any `tn3270_session_data` subrecord that uses the same *port_number* as this one (for the same *client_address* or a different one) must also have a value specified for this parameter, although the address need not be the same for all sessions.

Note: If you specify a local address for one or more sessions, this client record will not be displayed in the Motif administration program, so you cannot use that program to

view or manage it. You can still manage it using the command-line administration program, **snaadmin**, or a NOF application.

lu_name

Name of the display LU or LU pool that this session uses for connections requesting a generic display LU. This name is an 8-byte type-A character string. It must match the name of a type 0–3 display LU defined on this node, or an LU pool containing display LUs on this node.

If you specify an LU name, the client with the specified TCP/IP address can use only one generic display LU at a time through this TN server node. If you specify an LU pool, the client can use multiple generic display LU sessions, up to the number of LUs on this node that are available from the pool.

This parameter has no effect if the client requests access to a specific display LU.

ssl_enabled

Indicates whether this session uses Secure Sockets Layer (SSL) to access the server.

This parameter is reserved if you have not installed the additional software required to support SSL on the server. You can check this by using the **query_node_limits** command and checking the value of the *ssl_support* parameter.

Possible values are:

NO This session does not use SSL.

YES This session uses SSL.

YES_WITH_CLI_AUTH

This session uses SSL, and the TN Server requires it to use client authentication. The client must send a valid certificate (information identifying it as a valid client authorized to use the TN Server).

As well as checking that the certificate is valid, the TN Server may also need to check the certificate against a certificate revocation list on an external LDAP server, to ensure that the user's authorization has not been revoked. In this case, you also need to use **define_tn3270_ssl_ldap** to specify how to access this server.

Note:

1. If this session's *port_number* parameter indicates that it uses the Telnet daemon's TCP/IP port, do not use SSL for this session. If you use SSL on a session that uses the Telnet daemon's TCP/IP port, Telnet clients will not be able to use **telnet** to access the Communications Server for Linux computer while the node is active.
2. If you have large numbers of clients that use the same port, and are migrating them from non-SSL to SSL configuration, you can set up the configuration to accept both SSL and non-SSL connections on the same port

while the migration is in progress. See the *allow_ssl_timeout_to_nonssl* parameter below.

security_level
Indicates the SSL security level required for this session. The session will use the highest security level that both client and server can support; if the client cannot support the requested level of security or higher, the session will not be started.

If the *ssl_enabled* parameter is set to `NO`, this parameter is not used.

Possible values are:

SSL_AUTHENTICATE_MIN
Certificates must be exchanged; encryption is not required (but can be used if the client requests it).

SSL_AUTHENTICATE_ONLY
Certificates must be exchanged, but encryption will not be used. This option is typically used to avoid the overhead of encryption when the client is connecting across a secure intranet.

SSL_40_BIT_MIN
Use at least 40-bit encryption.

SSL_56_BIT_MIN
Use at least 56-bit encryption.

SSL_128_BIT_MIN
Use at least 128-bit encryption.

SSL_168_BIT_MIN
Use at least 168-bit encryption.

Note: Using encryption requires additional software to be installed with Communications Server for Linux; see *Communications Server for Linux Quick Beginnings* for more information. Depending on your location, you may not be able to use all the encryption levels listed because the software required to support them is not available in your country.

cert_key_label
The label identifying a certificate and key pair for use with SSL on this session. This must match a label specified when the SSL keyring database was set up; see *Communications Server for Linux Quick Beginnings* for more information.

To use the default SSL certificate and key pair, specified when the SSL keyring database was set up, do not specify this parameter.

allow_ssl_timeout_to_nonssl
This parameter does not apply if *ssl_enabled* is set to `NO`. Indicates whether non-SSL TN3270 clients can access the server using this session record even though it is configured to use SSL. Possible values are:

YES TN3270 clients not using SSL can access the server. There will be a 5–second delay on startup while the server waits for SSL negotiation to begin; after this, the server will assume that the client is not using SSL and revert to normal TN3270 communications.

NO Only TN3270 clients using SSL can access the server.

Note: This option is provided for migration purposes: if you have large numbers of clients that use the same port, and are migrating them from non-SSL to SSL configuration, you can set up the configuration to accept both SSL and non-SSL connections on the same port while the migration is in progress.

Allowing non-SSL clients to use SSL resources may be a security exposure, so this option is not intended for long-term use. You should set this parameter to YES only for brief periods while migration is in progress, and then set it to NO when migration is complete.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc

PARAMETER_CHECK

secondary_rc

UNKNOWN_CLIENT_ADDRESS
The name or alias specified by the *client_address* parameter could not be mapped to a fully qualified name.

CLIENT_ADDRESS_CLASH
The fully qualified name, resolved from the *client_address* parameter, matches one that has already been defined.

DUPLICATE_PORT_NUMBER
Another TN3270 access session record uses the same *port_number* parameter as this one, but the *listen_local_address* parameters are set inconsistently. The *listen_local_address* must be specified on all records with the same port number, or on none of them; it cannot be specified on one but not specified on another.

TCPIP_PORT_IN_USE
The TCP/IP port number specified by the *port_number* parameter cannot be used by TN server because it is already in use by a different program.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_tn3270_association

The **define_tn3270_association** command defines an association between a display LU and a printer LU. This association allows a TN3270E client to connect to the printer LU that is associated with a display LU without knowing the name of the printer LU. The **define_tn3270_association** command can be used to define a new association or to overwrite an existing association for a particular display LU.

Supplied Parameters

Parameter [define_tn3270_association]	Type	Length	Default
display_lu_name	character	8	
description	character	31	(null string)
printer_lu_name	character	8	

Supplied parameters are:

display_lu_name
: Specifies the name of the display LU to be associated with the printer specified by the *printer_lu_name* parameter. This name consists of 1–8 type-A characters.

  The specified display LU should be a display LU defined on the local node.

description
: An optional text string describing the association. Communications Server for Linux uses this string is for information only. It is stored in the node's configuration file and returned on the **query_tn3270_association** command.

printer_lu_name
: Name of the printer LU to be associated with the display LU specified by the *display_lu_name* parameter. This name consists of 1–8 type-A characters.

  The specified printer LU should be a printer LU defined on the local node.

  It is not possible for a single printer LU to be shared by two TN3270E emulators; no two TN3270 associations should specify the same printer LU.

  The printer LU should not be accessible as a generic printer LU; otherwise, it may not be available as an associated printer LU because it is already in use. Therefore, the associated printer LU should not be configured (directly or indirectly as a member of an LU pool) as the *printer_lu_name* in a **define_tn3270_access** command.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check
No parameter errors occur for this command.

State Check
No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions
Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

define_tn3270_defaults

The **define_tn3270_defaults** command defines TN3270 parameters used on all client sessions.

If you are using Secure Sockets Layer (SSL) client authentication, and checking clients against a certificate revocation list on an external LDAP server, you also need to configure details of how to access this server. To do this, use the **define_tn3270_ssl_ldap** command.

Supplied Parameters

Parameter	Type	Length	Default
[define_tn3270_defaults]			
force_responses	constant		NO
keepalive_method	constant		NONE
keepalive_interval	decimal		600

Supplied parameters are:

force_responses
Controls client responses on printer sessions. Possible values are:

YES Always request definite responses from the client printer sessions. Some 3270 emulators are unable to print large jobs if definite responses are not requested. If necessary, set *force_responses* to YES to avoid problems.

NO Request responses matching SNA traffic.

keepalive_method
Method for sending keep-alive messages. Keep-alive messages are messages sent to TN3270 clients when there is no other activity on the connection, to keep the TCP/IP connections to the clients active; this ensures that failed connections and clients can be detected. If there is no traffic at all on a TCP/IP connection, failure of the connection or of the client may never be detected, which wastes TN server resources and prevents LUs from being used for other sessions.

Possible values are:

NONE Do not send keep-alive messages.

NOP Send Telnet NOP messages.

TM Send Telnet DO TIMING-MARK messages.

keepalive_interval
Interval (in seconds) between consecutive keep-alive messages. The interval should be long enough to minimize network traffic, especially if there are

typically many idle client connections. The shorter the keep-alive interval, the quicker failures are detected, but the more network traffic is generated. If the keep-alive interval is too short and there are many clients, this traffic can be significant.

Typical values are in the range `600`–`7200` (10 minutes to 2 hours). The value `0` (zero) is not valid if *keepalive_method* is set to `NOP` or `TM`.

Because of the way TCP/IP operates, the keepalive interval that you configure is not the exact time that it will take for the server to recognize that a client has disappeared. The exact time depends on various factors, but will be no more than twice the configured timeout plus a few extra minutes (the exact number depends on how TCP/IP is configured).

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

define_tn3270_express_logon

The **define_tn3270_express_logon** command sets up the TN3270 Express Logon feature. This feature means that TN3270 client users who connect to Communications Server for Linux TN Server or TN Redirector using the Secure Sockets Layer (SSL) client authentication feature do not need to supply the user ID and password normally used for TN3270 security. Instead, their security certificate is checked against a Digital Certificate Access Server (DCAS) at the host, which supplies the required user ID and password.

Supplied Parameters

```
Parameter               Type          Length    Default
[define_tn3270_express_logon]
dcas_server             character     255
dcas_port               decimal                 8990
enabled                 constant                YES
```

Supplied parameters are:

dcas_server

The TCP/IP address of the host DCAS server that handles Express Logon authorization. This can be specified as any of the following.

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

If you use a name or alias, the Linux system 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). Fully-qualified names are not case-sensitive; for example, `Newbox.THIS.CO.UK` is equivalent to `newbox.this.co.uk`.

dcas_port
The TCP/IP port number used to access the DCAS server.

enabled Specifies whether the TN3270 Express Logon function is enabled. Possible values are:

YES The function is enabled, so TN3270 clients can access the host without needing to specify a user ID and password.

NO The function is not enabled, so TN3270 clients must specify a user ID and password.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_tn3270_ssl_ldap

The **define_tn3270_ssl_ldap** command defines how to access a certificate revocation list for use with the Secure Sockets Layer (SSL) client authentication feature. The revocation list is held on an external LDAP server, and contains details of individual TN3270 clients that are no longer authorized to use TN Server or TN Redirector (for example because the user's security information has been discovered by an unauthorized party, or because the user no longer works for the authorized organization).

If this feature is in use, a TN3270 client connecting to Communications Server for Linux TN Server or TN Redirector must supply a certificate (information identifying it as a valid client authorized to use the server). The server then checks this certificate against the revocation list to ensure that it is still valid.

This command can be used to define access to the LDAP server, to modify the access information (for example to change a user ID and password), or to specify that Communications Server for Linux does not use a revocation list on an external LDAP server.

The command must be issued to an inactive node; you cannot modify the LDAP server access information while the node is running.

Supplied Parameters

Parameter	Type	Length	Default
[define_tn3270_ssl_ldap]			
{tn3270_ssl_ldap_data}			
auth_type	constant		LOCAL_ONLY
ldap_addr	character	255	
ldap_port	decimal		
ldap_user	character	1024	
ldap_password	character	128	

Supplied parameters are:

auth_type
Specifies the type of authorization checking performed by the TN Server or TN Redirector. Possible values are:

LOCAL_ONLY
The server checks client certificates locally, but does not use an external certificate revocation list. The parameters *ldap_addr*—*ldap_password* are not used.

LOCAL_X500
The server checks certificates locally, and also checks against an external certificate revocation list. The remaining parameters on this command specify the location of this list.

ldap_addr
The TCP/IP address of the LDAP server that holds the certificate revocation list. This can be specified as any of the following.

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

If you use a name or alias, the Linux system 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). Fully-qualified names are not case-sensitive; for example, `Newbox.THIS.CO.UK` is equivalent to `newbox.this.co.uk`.

ldap_port
The TCP/IP port number used to access the LDAP server.

ldap_user
The user name used to access the certificate revocation list on the LDAP

server. Check with the system administrator of the LDAP server to determine how to specify this parameter.

ldap_password
The password used to access the certificate revocation list on the LDAP server. Check with the system administrator of the LDAP server to determine how to specify this parameter.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

INVALID_AUTH_TYPE
The *auth_type* parameter was not set to a valid value.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_tn_redirect

The **define_tn_redirect** command defines access details for a particular Telnet client (or default access details for all clients) using the TN Redirector feature of Communications Server for Linux. It can be used to define a new client, or to modify the existing definition. (To define access details for a client using TN3270 Server, use **define_tn3270_access**.)

Supplied Parameters

Parameter	Type	Length	Default
[define_tn_redirect]			
default_record	constant		YES
client_address	character	256	(null string)
client_port	decimal		
listen_local_address	character	45	(null string)
cli_conn_ssl_enabled	constant		NO
cli_conn_security_level	constant		SSL_AUTHENTICATE_MIN
cli_conn_cert_key_label	character	80	(null string)
host_address	character	255	(null string)
host_port	decimal		

```
serv_conn_ssl_enabled    constant                NO
serv_conn_security_level constant                SSL_AUTHENTICATE_MIN
serv_conn_cert_key_label character    80         (null string)
description              character    31         (null string)
```

Supplied parameters are:

default_record
: Specifies whether this command defines a default record, which will be used by any Telnet user not explicitly identified by a TCP/IP address. If a user attempts to contact the TN Redirector node, and the user's TCP/IP address does not match any TN Redirector record in the configuration but there is a default record defined for the port number used by the client, the parameters from this record will be used. Possible values are:

  YES This command defines a default record. The *client_address* parameter is not used.

  NO This command defines a normal TN Redirector user record.

  A default record provides access to the TN Redirector function for any Telnet user that can determine the TCP/IP address of the computer where the TN server is running. To restrict the use of TN Redirector to a specific group of users, either do not include the default record, or leave it with no host address 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 Redirector users, and leave them with no host address configured.

client_address
: The TCP/IP address of the computer on which the Telnet program runs. This can be specified as any of the following.

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

  If you use a name or alias, the following restrictions apply:

  - The Linux system 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.
  - Fully-qualified names are not case-sensitive; for example, `Newbox.THIS.CO.UK` is equivalent to `newbox.this.co.uk`.

client_port
: The number of the server TCP/IP port that the Telnet program uses to access the TN Redirector node.

  If the *default_record* parameter specifies that this is a default TN Redirector access record, this parameter must not match the port address used by a default TN3270 Server access record (defined using **define_tn3270_access**). You can define only one of the two types of default record for each port number.

If two or more **define_tn_redirect** commands use the same *client_port* (for the same *client_address* or a different one), the *listen_local_address* parameter must be specified on all of them or none of them; you cannot specify it on some sessions but leave it unspecified on others.

listen_local_address
The address on the local TN Server computer to which TN3270 clients will connect. This parameter is optional.

- If TN3270 clients are to be able to connect on any local address, or if there is only one valid local address on the TN Server, do not specify this parameter. In this case, any **define_tn_redirect** command that uses the same *port_number* as this one (for the same *client_address* or a different one) must also leave this parameter unspecified.
- If you need to restrict TN3270 clients to a particular local address, specify it as one of the following:
  - An IPv4 dotted-decimal address (such as `193.1.11.100`).
  - An IPv6 colon-hexadecimal address (such as `2001:0db8:0000:0000:0000:0000:1428:57ab` or `2001:db8::1428:57ab`).

  In this case, any **define_tn_redirect** command that uses the same *port_number* as this one (for the same *client_address* or a different one) must also have a value specified for this parameter, although the address need not be the same for all sessions.

Note: If you specify a local address for one or more sessions, this client record will not be displayed in the Motif administration program, so you cannot use that program to view or manage it. You can still manage it using the command-line administration program, **snaadmin**, or a NOF application.

cli_conn_ssl_enabled
Indicates whether the client uses Secure Sockets Layer (SSL) to access the TN Redirector.

This parameter is reserved if you have not installed the additional software required to support SSL on the server. You can check this by using the **query_node_limits** command and checking the value of the *ssl_support* parameter.

Possible values are:

NO The client does not use SSL.

YES The client uses SSL.

YES_WITH_CLI_AUTH
The client uses SSL, and the TN Server requires it to use client authentication. The client must send a valid certificate (information identifying it as a valid client authorized to use the TN Server).

As well as checking that the certificate is valid, the TN Server may also need to check the certificate against a certificate revocation list on an external LDAP server, to ensure that the user's authorization has not been revoked. In this case, you also need to use **define_tn3270_ssl_ldap** to specify how to access this server.

cli_conn_security_level
Indicates the SSL security level required for the client connection on this session. The session will use the highest security level that both client and

server can support; if the client cannot support the requested level of security or higher, the session will not be started.

If the *cli_conn_ssl_enabled* parameter is set to NO, this parameter is not used.

Possible values are:

SSL_AUTHENTICATE_MIN
Certificates must be exchanged; encryption is not required (but can be used if the client requests it).

SSL_AUTHENTICATE_ONLY
Certificates must be exchanged, but encryption will not be used. This option is typically used to avoid the overhead of encryption when the client is connecting across a secure intranet.

SSL_40_BIT_MIN
Use at least 40-bit encryption.

SSL_56_BIT_MIN
Use at least 56-bit encryption.

SSL_128_BIT_MIN
Use at least 128-bit encryption.

SSL_168_BIT_MIN
Use at least 168-bit encryption.

Note: Using encryption requires additional software to be installed with Communications Server for Linux; see *Communications Server for Linux Quick Beginnings* for more information. Depending on your location, you may not be able to use all the encryption levels listed because the software required to support them is not available in your country.

cli_conn_cert_key_label
The label identifying a certificate and key pair for use with SSL on the client session. This must match a label specified when the SSL keyring database was set up; see *Communications Server for Linux Quick Beginnings* for more information.

If the *cli_conn_ssl_enabled* parameter is set to NO, this parameter is not used.

To use the default SSL certificate and key pair, specified when the SSL keyring database was set up, do not specify this parameter.

host_address
The TCP/IP address of the host computer with which the client communicates. This can be specified as any of the following.

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

If you use a name or alias, the Linux system 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). Fully-qualified names are not case-sensitive; for example, Newbox.THIS.CO.UK is equivalent to newbox.this.co.uk.

host_port
The number of the TCP/IP port that the TN server node uses to access the host.

serv_conn_ssl_enabled
Indicates whether the TN server uses Secure Sockets Layer (SSL) to access the host on behalf of this client.

This parameter is reserved if you have not installed the additional software required to support SSL on the server. You can check this by using the **query_node_limits** command and checking the value of the *ssl_support* parameter.

Possible values are:

NO The host does not use SSL.

YES The host uses SSL.

serv_conn_security_level
Indicates the SSL security level required for the host connection on this session. The session will use the highest security level that both the host and Communications Server for Linux can support; if the host cannot support the requested level of security or higher, the session will not be started.

If the *serv_conn_ssl_enabled* parameter is set to NO, this parameter is not used.

Possible values are:

SSL_AUTHENTICATE_MIN
Certificates must be exchanged; encryption is not required (but can be used if the host requests it).

SSL_AUTHENTICATE_ONLY
Certificates must be exchanged, but encryption will not be used. This option is typically used to avoid the overhead of encryption when the host connection is across a secure intranet.

SSL_40_BIT_MIN
Use at least 40-bit encryption.

SSL_56_BIT_MIN
Use at least 56-bit encryption.

SSL_128_BIT_MIN
Use at least 128-bit encryption.

SSL_168_BIT_MIN
Use at least 168-bit encryption.

Note: Using encryption requires additional software to be installed with Communications Server for Linux; see *Communications Server for Linux Quick Beginnings* for more information. Depending on your location, you may not be able to use all the encryption levels listed because the software required to support them is not available in your country.

serv_conn_cert_key_label
The label identifying a certificate and key pair for use with SSL on the host

session. This must match a label specified when the SSL keyring database was set up; see *Communications Server for Linux Quick Beginnings* for more information.

If the *serv_conn_ssl_enabled* parameter is set to NO, this parameter is not used.

To use the default SSL certificate and key pair, specified when the SSL keyring database was set up, do not specify this parameter.

description
An optional text string (0–31 characters followed by a null character). The string is for information only; it is stored in the configuration file and returned on a **query_tn_redirect_def** command, but Communications Server for Linux does not make use of it. You can use it to store additional information to help distinguish between users.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

UNKNOWN_CLIENT_ADDRESS
The name or alias specified by the *client_address* parameter could not be mapped to a fully qualified name.

CLIENT_ADDRESS_CLASH
The combination of port number and fully qualified name (resolved from the *client_address* parameter) matches one that has already been defined.

DUPLICATE_PORT_NUMBER
Another TN Redirector record uses the same *client_port* parameter as this one, but the *listen_local_address* parameters are set inconsistently. The *listen_local_address* must be specified on all records with the same port number, or on none of them; it cannot be specified on one but not specified on another.

TCPIP_PORT_IN_USE
The TCP/IP port number specified by the *client_port* or *host_port* parameter cannot be used by TN Redirector because it is already in use by a different program.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_tp

The **define_tp** command provides information that Communications Server for Linux needs to start a TP as a result of an incoming Attach from a partner LU. This command can be used to define a TP for the first time or to modify one or more parameters on a previously defined TP.

The standard parameters for invoked TPs are defined in the invokable TP data file; for more information, refer to the *Communications Server for Linux Administration Guide*. The **define_tp** command is required only if you need to specify additional parameters that cannot be set in the TP data file. These additional parameters restrict the use of options that specify conversation security, confirm synchronization, and specify conversation type (mapped or basic) for the TP, or restrict the number of instances of the TP that can be running at any time.

Supplied Parameters

Parameter name [define_tp]	Type	Length	Default
tp_name	character	64	
description	character	31	(null string)
list_name	character	14	(null string)
conv_type	constant		EITHER
security_rqd	constant		NO
sync_level	constant		SYNCPT_NEGOTIABLE
enabled	constant		YES
pip_allowed	constant		YES
tp_instance_limit	decimal		0
incoming_alloc_timeout	decimal		0

Supplied parameters are:

tp_name
: Name of the TP to be defined.

description
: A text string describing the TP. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_tp_definition** and **query_tp** commands.

list_name
: Name of the security access list used by this TP (defined using the **define_security_access_list** command). This parameter restricts the TP so that only the users named in the specified list can allocate conversations with it. If you specify a security access list, the *security_rqd* parameter must be set to YES.

  To specify that the TP is available for use by any user, do not specify this parameter.

conv_type
: Specifies the type or types of conversation supported by this TP. Possible values are:

  BASIC The TP supports only basic conversations.

  MAPPED The TP supports only mapped conversations.

EITHER The TP supports both basic and mapped conversations.

security_rqd
Specifies whether conversation security information is required to start the TP. Possible values are:

YES A user ID and password are required to start the TP.

NO No security information is required to start the TP.

sync_level
Specifies the values of synchronization level supported by the TP. Possible values are:

NONE The TP supports only the *sync_level* value of NONE.

CONFIRM_SYNC_LEVEL
The TP supports only the *sync_level* value of CONFIRM.

EITHER The TP supports both the *sync_level* values NONE and CONFIRM.

SYNCPT_REQUIRED
The TP supports only the *sync_level* value of SYNCPT (sync point is required).

SYNCPT_NEGOTIABLE
The TP supports all three *sync_level* values of NONE, CONFIRM, and SYNCPT.

enabled Specifies whether the TP can be attached successfully. Possible values are:

YES TP can be attached.

NO TP cannot be attached.

pip_allowed
Specifies whether the TP can receive program initialization parameters (PIP). Possible values are:

YES TP can receive PIP.

NO TP cannot receive PIP.

tp_instance_limit
Limit on the number of instances of this TP that can be active at any one time. Specify a value in the range 1–65,535, or 0 (zero) for no limit.

incoming_alloc_timeout
Specifies the number of seconds that an incoming Attach will be queued waiting for a RECEIVE_ALLOCATE. The value 0 (zero) implies that there is no timeout; the incoming Attach will be queued indefinitely.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

SYSTEM_TP_CANT_BE_CHANGED
The specified TP name is the name of a TP used internally by Communications Server for Linux; you cannot define or modify a TP with this name.

INVALID_CONV_TYPE
The *conv_type* parameter was not set to a valid value.

INVALID_SYNC_LEVEL
The *sync_level* parameter was not set to a valid value.

INVALID_ENABLED
The *enabled* parameter was not set to a valid value.

INVALID_PIP_ALLOWED
The *pip_allowed* parameter was not set to a valid value.

State Check
If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

SECURITY_LIST_NOT_DEFINED
The *security_list_name* parameter did not match any defined security list name.

Other Conditions
Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_tp_load_info

The **define_tp_load_info** command is used to define or change a TP load information entry..

Supplied Parameters

Parameter name	Type	Length	Default
[define_tp_load_info]			
tp_name	character	64	
lualias	character	8	
description	character	31	
path	character	255	(null string)
arguments	character	255	(null string)
type	constant		QUEUED
timeout	decimal		-1
userid	character	64	
group	character	64	(null string)
stdin	character	255	/dev/null

```
stdout                    character    255       /dev/null
stderr                    character    255       /dev/null
env                       character    255       (null string)
```

(0–64 env entries can be included)

Supplied parameters are:

tp_name
The TP name of the TP load info entry to be defined.

lualias The LU alias of the TP load info entry to be defined.

Note: This parameter can be used only if the TP is an APPC TP. If the TP is a CPI-C application, do not specify this parameter. CPI-C does not support accepting incoming Attaches from a particular local LU; specifying an LU alias (even a blank LU alias) for a CPI-C application will cause errors in routing the incoming Attach to the TP.

description
Optional text string describing the TP load info.

path The full path name of the TP executable.

arguments
Command-line arguments required by the TP. These arguments are separated with spaces.

type Specifies the TP type. Possible values are:

QUEUED The TP is a queued TP.

QUEUED-BROADCAST
The TP is a broadcast queued TP.

NON-QUEUED
The TP is a nonqueued TP.

timeout
Timeout in seconds after the TP is loaded. Specify a value in the range `0–65,535`. The value `-1` indicates an infinite timeout.

userid User ID required to access and run the TP.

group Group ID required to access and run the TP.

stdin Full path name of standard input file or device.

stdout Full path name of standard output file or device.

stderr Full path name of standard error file or device.

env Environment variables required by the TP in the form *VARIABLE = VALUE*. For more information about environment variables that the TP may require, see Appendix C, “Environment Variables,” on page 599.

If the TP is a CPI-C application, note that you cannot set the environment variable `APPCLLU` using this parameter. The local LU cannot be specified in the TP load information for an automatically-loaded CPI-C application.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_TP_TYPE
The *type* parameter was not set to a valid value.

INVALID_TP_NAME
The *tp_name* parameter specified did not match the name of a defined TP.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_tr_dlc, define_ethernet_dlc

The **define_tr_dlc** command defines a new Token Ring DLC. In addition, you can use this command to modify an existing DLC, if the DLC is not currently active. However, you cannot change the negotiable link support for an existing DLC; this parameter can be specified only when creating a new DLC.

The **define_ethernet_dlc** command defines a new Ethernet DLC. It can also be used to modify an existing DLC, if the DLC is not currently active. The parameters and defaults are the same as for **define_tr_dlc**, except where noted.

Supplied Parameters

Parameter name	Type	Length	Default
[define_tr_dlc], [define_ethernet_dlc]			
dlc_name	character	8	
description	character	31	(null string)
neg_ls_supp	constant		YES
adapter_number	decimal		0
initially_active	constant		YES

The following parameter is used only for Ethernet DLCs:

lan_type	constant	802_3_DIX

Supplied parameters are:

dlc_name
Name of the DLC to be defined. This name is a character string using any locally displayable characters.

description
A text string describing the DLC. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_dlc** command.

neg_ls_supp
Specifies whether the DLC supports negotiable link stations. You cannot change the negotiable link support for an existing DLC; this parameter can be specified only when creating a new DLC. Possible values are as follows:

YES Negotiable link stations are supported; link stations using this DLC can be primary, secondary, or negotiable.

NO Negotiable link stations are not supported; link stations using this DLC must be primary or secondary.

adapter_number
Adapter number used by the DLC.

If the server contains more than one adapter card for this DLC type, specify `0` for the first card, `1` for the second card, and so on. Otherwise, set this parameter to `0` (zero).

initially_active
Specifies whether this DLC is automatically started when the node is started. Possible values are:

YES The DLC is automatically started when the node is started.

NO The DLC is automatically started only if a port or LS that uses it is defined as initially active; otherwise, it must be manually started.

The following parameter is used only for Ethernet:

lan_type
Type of Ethernet network. Possible values are:

802_3 IEEE 802.3

DIX DIX

802_3_DIX
Either IEEE 802.3 or DIX

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

INVALID_DLC_NAME
The supplied *dlc_name* parameter contained a character that was not valid.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

DLC_ACTIVE
The *neg_ls_supp* parameter cannot be modified because the DLC is currently active.

INVALID_DLC_TYPE
You cannot change the negotiable link support for an existing DLC. This parameter can be specified only when creating a new DLC.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

define_tr_ls, define_ethernet_ls

The **define_tr_ls** command is used to define a new Token Ring link station (LS) or modify an existing one. Before issuing this command, you must define the port that this LS uses.

The **define_ethernet_ls** command is used to define a new Ethernet link station (LS) or modify an existing one. Before issuing this command, you must define the port that this LS uses. The parameters and defaults are the same as for **define_tr_ls**, except where noted.

You cannot use this command to modify the port used by an existing LS; the *port_name* specified on the command must match the previous definition of the LS. The LS can be modified only if it is not started.

Supplied Parameters

Parameter name	Type	Length	Default
[define_tr_ls], [define_ethernet_ls]			
ls_name	character	8	
description	character	31	(null string)
port_name	character	8	
adj_cp_name	character	17	(null string)
adj_cp_type	constant		LEARN_NODE
mac_address	hex array	6	(null string)
lsap_address	hex number		0x04
auto_act_supp	constant		NO
tg_number	decimal		0
limited_resource	constant		NO
solicit_sscp_sessions	constant		NO
pu_name	character	8	(taken from ls_name)
disable_remote_act	constant		NO
dspu_services	constant		NONE
dspu_name	character	8	(taken from ls_name)
dlus_name	character	17	(null string)
bkup_dlus_name	character	17	(null string)
hpr_supported	constant		
hpr_link_lvl_error	constant		
link_deact_timer	decimal	30	
default_nn_server	constant		NO
ls_attributes	constant		SNA
adj_node_id	hex array	4	
local_node_id	hex array	4	
cp_cp_sess_support	constant		YES
use_default_tg_chars	constant		NO
effect_cap	decimal		16000000 (TR) 865075200 (Ethernet, Linux on System z) 157286400 (Ethernet, other Linux variants)
connect_cost	decimal		0
byte_cost	decimal		0
security	constant		SEC_NONSECURE
prop_delay	constant		PROP_DELAY_LAN
user_def_parm_1	decimal		0
user_def_parm_2	decimal		0
user_def_parm_3	decimal		0
target_pacing_count	decimal		7
max_send_btu_size	decimal		4105 (TR) 1033 (Ethernet)
ls_role	constant		USE_PORT_DEFAULTS
max_ifrm_rcvd	decimal		0
dlus_retry_timeout	decimal		0
dlus_retry_limit	decimal		0
conventional_lu_compression	constant		NO
branch_link_type	constant		UPLINK (used only if this node is BrNN)
adj_brnn_cp_support	constant		ALLOWED (used only if this node is BrNN)
initially_active	constant		NO
react_timer	decimal		30
react_timer_retry	decimal		65535
restart_on_normal_deact	constant		NO
xid_timer	decimal		10
xid_timer_retry	decimal		5
test_timeout	decimal		10
test_timer_retry	decimal		5
ack_timeout	decimal		5000
p_bit_timeout	decimal		5000
t2_timeout	decimal		100
rej_timeout	decimal		10
busy_state_timeout	decimal		30
idle_timeout	decimal		30
max_retry	decimal		3
dddlu_offline_supported	constant		NO

Supplied parameters are:

ls_name
Name of the link station to be defined.

description
A text string describing the LS. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_ls**, **query_pu**, and **query_downstream_pu** commands.

port_name
Name of the port associated with this link station. This name must match the name of a defined port.

adj_cp_name
Fully qualified name of the adjacent CP for this LS. Specify 3–17 type-A characters consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character CP name. This parameter is used in the following ways:

- If the *adj_cp_type* parameter is set to `NETWORK_NODE` or `END_NODE`, and preassigned TG numbers are being used, set this parameter to the CP name defined at the adjacent node; if the adjacent node sends a CP name during XID exchange, it will be checked against this value.
- If *adj_cp_type* is set to `BACK_LEVEL_LEN_NODE`, Communications Server for Linux uses this value only as an identifier; set it to any string that does not match other CP names defined at this node.
- If *adj_cp_type* is set to any other value, or if preassigned TG numbers are not being used, there is no need to specify this parameter; Communications Server for Linux checks the CP name only if one is specified.

adj_cp_type
Adjacent node type.

If the adjacent node is an APPN node and preassigned TG numbers are not being used, this parameter is usually set to `LEARN_NODE`, indicating that the node type is unknown. Communications Server for Linux will determine the type during XID exchange.

If preassigned TG numbers are being used, you must specify the node type explicitly. You can also specify the type as an additional security check if preassigned TG numbers are not being used. In this case, Communications Server for Linux will reject a connection attempt from the adjacent node if its node type does not match the one specified for this parameter. Possible values are:

`LEARN_NODE`
The adjacent node type is unknown; Communications Server for Linux will determine the type during XID exchange.

`END_NODE`
The adjacent node is an End Node, a Branch Network Node acting as an End Node from the local node's perspective, or a LEN node that includes the Network Name CV in its XID3.

`NETWORK_NODE`
The adjacent node is a Network Node, or a Branch Network Node acting as a Network Node from the local node's perspective.

If the adjacent node is not an APPN node, possible values are:

BACK_LEVEL_LEN_NODE
The adjacent node is one that does not include the Network Name control vector in its XID3.

HOST_XID3
The adjacent node is a host node; Communications Server for Linux responds to a polling XID from the node with a format 3 XID.

HOST_XID0
The adjacent node is a host node; Communications Server for Linux responds to a polling XID from the node with a format 0 XID.

DSPU_XID
The adjacent node is a downstream PU; Communications Server for Linux includes XID exchange in link activation. The *dspu_name* and *dspu_services* parameters must also be set.

DSPU_NOXID
The adjacent node is a downstream PU; Communications Server for Linux does not include XID exchange in link activation. The *dspu_name* and *dspu_services* parameters must also be set.

If you want to run independent LU 6.2 traffic over this LS, you must set the *adj_cp_type* parameter to `LEARN_NODE`, `END_NODE`, `NETWORK_NODE`, or `BACK_LEVEL_LEN_NODE`.

mac_address
MAC address of the adjacent node.

If you need to define a non-selective listening LS (one that can be used only for incoming calls, but can have LUs defined on it to support dependent LU traffic), do not specify this parameter. The LS can then be used to receive incoming calls from any remote link station, but cannot be used for outgoing calls. There is no need to define a non-selective listening LS if only independent LU traffic is used, because an LS for independent LU traffic can be set up dynamically when required.

You will probably need to reverse the bit order of the bytes in the MAC address if the local and adjacent nodes are on LANs of different types (one Ethernet, the other Token Ring) connected by a bridge. For more information, see "Bit Ordering in MAC Addresses" on page 225. If the two nodes are on the same LAN, or on LANs of the same type connected by a bridge, no change in bit order is required.

lsap_address
Local SAP address of the adjacent node. Specify a multiple of `0x04` in the range `0x04–0xEC`.

auto_act_supp
Specifies whether the link can be automatically activated when required by a session. Possible values are:

YES The link can be automatically activated.

The reactivation timer parameters are ignored. If the LS fails, Communications Server for Linux does not attempt to reactivate it until a dependent LU application that was using the session attempts to restart the session; an LS used by independent LUs is not reactivated by Communications Server for Linux and must be manually restarted.

The following restrictions also apply:

- If the link is to an APPN node, the LS must have a preassigned TG number defined (see the *tg_number* parameter), and *cp_cp_sess_support* must be set to NO.
- If either the local node or the adjacent node is an end node, the LS must also be defined as automatically activated at the adjacent node.

NO The link cannot be automatically activated.

tg_number
Preassigned TG number. This parameter is used only if the adjacent node is an APPN node (*adj_cp_type* is either NETWORK_NODE or END_NODE); it is ignored otherwise.

This TG number is used to represent the link when the link is activated. The node does not accept any other number from the adjacent node during activation of this link; if the adjacent node is using preassigned TG numbers, the same TG number must be defined by the adjacent node on the adjacent link station.

If the local node is a LEN node, or if the adjacent node is a LEN node and the link is automatically activated, set the TG number to 1. Otherwise, specify a number in the range 1–20, or 0 (zero) to indicate that the TG number is not preassigned and is negotiated when the link is activated.

If a preassigned TG number is defined, the *adj_cp_name* parameter must also be defined, and the *adj_cp_type* parameter must be set to either END_NODE or NETWORK_NODE.

limited_resource
Specifies whether this link station is a limited resource and is automatically deactivated when there are no sessions using the link. Link stations on a nonswitched port cannot be configured as limited resource. Possible values are:

NO The link is not a limited resource and is not automatically deactivated.

NO_SESSIONS
The link is a limited resource and is automatically deactivated when no active sessions are using it.

INACTIVITY
The link is a limited resource and is automatically deactivated when no active sessions are using it, or when no data has flowed on the link for the time period specified by the *link_deact_timer* parameter.

- If no SSCP-PU session is active across the link, the node deactivates the link immediately.
- If an SSCP-PU session is active but no traffic has flowed for the specified time period, the node sends REQDISCONT(normal) to the host. The host is then responsible for deactivating all LUs and the PU, at which time the local node will deactivate the link. However, the host may not deactivate LUs with active PLU-SLU sessions; in this case, the link remains active until all these sessions are deactivated (for example by the user logging out). This behavior can be changed by using options in the *ptf* parameter of the **start_node** command.

A limited resource link station can be configured for CP-CP session support by setting this parameter to NO_SESSIONS and *cp_cp_sess_support* to YES. In this case, if CP-CP sessions are brought up over the link, Communications Server for Linux does not treat the link as a limited resource (and therefore does not deactivate it).

solicit_sscp_sessions
Specifies whether to request the adjacent node to initiate sessions between the SSCP and the local CP and dependent LUs. This parameter is used only if the adjacent node is an APPN node (*adj_cp_type* is either NETWORK_NODE or END_NODE); it is ignored otherwise. If the adjacent node is a host (*adj_cp_type* is either HOST_XID3 or HOST_XID0), Communications Server for Linux always requests the host to initiate SSCP sessions.

Possible values are:

YES Request the adjacent node to initiate SSCP sessions.

NO Do not request the adjacent node to initiate SSCP sessions.

If the adjacent node is an APPN node and *dspu_services* is set to a value other than NONE, this parameter must be set to NO.

pu_name
Name of the local PU that uses this link. This parameter is required only if *adj_cp_type* is set to HOST_XID3 or HOST_XID0, or if *solicit_sscp_sessions* is set to YES; it is ignored otherwise. This name is a type-A character string starting with a letter.

You cannot change the PU name on an LS that is already defined.

If the PU name is required and you do not specify it, the default is the same as the LS name. To ensure that this name is a valid type-A character string, Communications Server for Linux converts it to uppercase; if the string begins with a numeric character, this character is either removed or preceded by the characters "PU."

disable_remote_act
Specifies whether to prevent activation of the LS by the remote node. Possible values are:

YES The LS can be activated only by the local node; if the remote node attempts to activate it, Communications Server for Linux will reject the attempt.

NO The LS can be activated by the remote node.

dspu_services
Specifies the services that the local node provides to the downstream PU across this link. This parameter is used only if the adjacent node is a downstream PU or an APPN node with *solicit_sscp_sessions* set to NO; it is reserved otherwise. Possible values are:

PU_CONCENTRATION
Local node provides SNA gateway for the downstream PU. The local node must be defined to support SNA gateway.

DLUR Local node provides DLUR services for the downstream PU. The local node must be defined to support DLUR. (DLUR is not supported on end node.)

NONE Local node provides no services for the downstream PU.

dspu_name
Name of the downstream PU. The name is a type-A character string starting with a letter. To ensure that this name is a valid type-A character string, Communications Server for Linux converts it to uppercase; if the string begins with a numeric character, this character is either removed or preceded by the characters "PU."

This parameter is reserved except when both of the following conditions are true:

- The *solicit_sscp_sessions* parameter is set to NO
- The *dspu_services* parameter is set to PU_CONCENTRATION or DLUR

If both of these conditions are true and you do not specify a value for *dspu_name*, the default is the same as the LS name.

If the downstream PU is used for DLUR, this name should match the PU name configured on the host. (Communications Server for Linux sends both the PU name and PU ID to the host to identify the PU. The host normally identifies the PU by its PU name, or by the PU ID if it cannot find a matching PU name.)

dlus_name
Name of the DLUS node from which DLUR solicits SSCP services when the link to the downstream node is activated. This parameter is reserved if *dspu_services* is not set to DLUR.

The name is a type-A character string, consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character DLUS name.

To specify the global default DLUS defined using **define_dlur_defaults**, do not specify this parameter. If this parameter is not specified and there is no global default DLUS, then DLUR will not initiate SSCP contact when the link is activated.

bkup_dlus_name
Name of the backup DLUS node from which DLUR solicits SSCP services if the node specified by *dlus_name* is not active. This parameter is reserved if *dspu_services* is not set to DLUR.

The name is a type-A character string, consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character backup DLUS name.

To specify the global backup default DLUS defined using **define_dlur_defaults**, do not specify this parameter.

hpr_supported
Specifies whether HPR is supported on this link. This parameter must be set to NO unless the *adj_cp_type* parameter indicates that the link connects to an APPN node. Possible values are:

YES HPR is supported on this link.

NO HPR is not supported on this link.

hpr_link_lvl_error
Specifies whether HPR traffic should be sent on this link using link-level error recovery. This parameter is ignored unless *hpr_supported* is set to YES. Possible values are:

YES HPR traffic should be sent on this link using link-level error recovery.

NO HPR traffic should not be sent on this link using link-level error recovery.

link_deact_timer
Limited resource link deactivation timer, in seconds. A limited resource link is automatically deactivated if no data flows over the link for the time specified by this parameter. This parameter is not used if *limited_resource* is set to any value other than INACTIVITY.

The minimum value is 5; values in the range 1–4 will be interpreted as 5.

The value 0 (zero) indicates one of the following:

- If the *hpr_supported* parameter is set to YES, the default deactivation timer value of 30 is used.
- If the *hpr_supported* parameter is set to NO, no timeout is used (the link is not deactivated, as if *limited_resource* were set to NO).

default_nn_server
For an end node this parameter specifies whether the link station being defined supports CP-CP sessions to a network node that can act as the local node's network node server. When the local node has no CP-CP sessions to a network node server and needs to establish them, the local node checks this parameter on its defined link stations to find a suitable LS to activate. This enables you to specify which adjacent NNs are suitable to act as the NN server (for example, to avoid using NNs that are accessed by expensive or slow links).

Possible values are:

YES This link supports CP-CP sessions to a network node that can act as the local node's NN server; the local node can automatically activate this link if it needs to contact an NN server. The *cp_cp_sess_support* parameter must be set to YES.

NO This link does not support CP-CP sessions to a network node that can act as the local node's NN server; the local node cannot automatically activate this link if it needs to contact an NN server.

If the local node is not an end node, this parameter is ignored.

ls_attributes
Attributes of the remote system with which Communications Server for Linux is communicating.

Specify SNA unless you are communicating with a host of one of the other following types. Possible values are:

SNA Standard SNA host

FNA Fujitsu Network Architecture (VTAM-F) host

HNA Hitachi Network Architecture host

SUPPRESS_CP_NAME
Suppress the CP name associated with the remote node. Use a + character to combine this value with SNA, FNA, or HNA.

If *adj_cp_type* is set to BACK_LEVEL_LEN_NODE, and the remote LEN node associated with this LS cannot accept the Network Name CV in the format

3 XID it receives, use a + character to combine the value SNA, FNA, or HNA with SUPPRESS_CP_NAME (for example, SNA+SUPPRESS_CP_NAME).

If *adj_cp_type* is set to any other value, the option SUPPRESS_CP_NAME is ignored.

adj_node_id
Node ID of adjacent node. This ID is a 4-byte hexadecimal string, consisting of a block number (three hexadecimal digits) and a node number (five hexadecimal digits). To disable node ID checking, do not specify this parameter.

local_node_id
Node ID sent in XIDs on this LS. This ID is a 4-byte hexadecimal string, consisting of a block number (three hexadecimal digits) and a node number (five hexadecimal digits). To use the node ID specified in the *node_id* parameter on the **define_node**, do not specify this parameter.

cp_cp_sess_support
Specifies whether CP-CP sessions are supported. This parameter is valid only if the adjacent node is an end node or a network node (*adj_cp_type* is NETWORK_NODE, END_NODE, or LEARN_NODE); it is ignored otherwise. If both the local node and the adjacent node are network nodes, this parameter should be set to YES in order to use APPN functions between these nodes.

Possible values are:

YES CP-CP sessions are supported.

NO CP-CP sessions are not supported.

use_default_tg_chars
Specifies whether to use the default TG characteristics supplied on **define_tr_port** / **define_ethernet_port** . The TG characteristics apply only if the link is to an APPN node; this parameter, and *effect_cap* through *user_def_parm_3* parameters are ignored otherwise. Possible values are:

YES Use the default TG characteristics; ignore *effect_cap* through *user_def_parm_3* parameters on this command.

NO Use *effect_cap* through *user_def_parm_3* parameters on this command.

effect_cap
A decimal value representing the line speed in bits per second.

For an Ethernet link, ensure that you set this parameter to the true 'effective capacity' of the link, including any step-downs or bottlenecks in the path, and not just to the theoretical capacity of the adapter used by the link. For example, a GigE adapter may be capable of processing one gigabit, but if the link goes through an ethernet switch to a target box that uses FastEthernet you should specify 100MBps or less.

connect_cost
Cost per connect time. Valid values are integer values in the range 0–255, where 0 is the lowest cost per connect time and 255 is the highest cost per connect time.

byte_cost
Cost per byte. Valid values are integer values in the range 0–255, where 0 is the lowest cost per byte and 255 is the highest cost per byte.

security
Security level of the network. Possible values are:

SEC_NONSECURE
No security.

SEC_PUBLIC_SWITCHED_NETWORK
Data is transmitted over a public switched network.

SEC_UNDERGROUND_CABLE
Data is transmitted over secure underground cable.

SEC_SECURE_CONDUIT
Data is transmitted over a line in a secure conduit that is not guarded.

SEC_GUARDED_CONDUIT
Data is transmitted over a line in a conduit that is protected against physical tapping.

SEC_ENCRYPTED
Data is encrypted before transmission over the line.

SEC_GUARDED_RADIATION
Data is transmitted over a line that is protected against physical and radiation tapping.

prop_delay
Propagation delay (the time that a signal takes to travel the length of the link). Specify one of the following values, according to the type of link:

PROP_DELAY_MINIMUM
Minimum propagation delay.

PROP_DELAY_LAN
Delay is less than .5 microseconds (typical for a LAN).

PROP_DELAY_TELEPHONE
Delay is in the range .5–50 microseconds (typical for a telephone network).

PROP_DELAY_PKT_SWITCHED_NET
Delay is in the range 50–250 microseconds (typical for a packet-switched network).

PROP_DELAY_SATELLITE
Delay is greater than 250 microseconds (typical for a satellite link).

PROP_DELAY_MAXIMUM
Maximum propagation delay.

user_def_parm_1 **through** *user_def_parm_3*
User-defined parameters, that you can use to include other TG characteristics not covered by the previous parameters. Each of these parameters must be set to a value in the range `0`–`255`.

target_pacing_count
Indicates the desired pacing window size. Specify a value in the range `1`–`32,767`.

max_send_btu_size
Maximum BTU size that can be sent from this link station. This value is used to negotiate the maximum BTU size that a pair of link stations can

use to communicate with each other. The value includes the length of the TH and RH (total 9 bytes), as well as the RU. Specify a value in the range 265–65535.

ls_role Link station role. This parameter is usually set to USE_PORT_DEFAULTS, specifying that the LS role is to be taken from the definition of the port that owns this LS.

If you need to override the port's LS role for an individual LS, specify one of the following values:

LS_PRI Primary

LS_SEC Secondary

LS_NEG Negotiable

max_ifrm_rcvd
Maximum number of I-frames that can be received by the local link stations before an acknowledgment is sent. Specify in the range 1–127.

dlus_retry_timeout
The interval in seconds between the second and subsequent attempts to contact the DLUS specified by the *dlus_name* and *bkup_dlus_name* parameters. The interval between the first and second attempts is always 1 second.

Specify a value in the range 0– 65,535. If you specify 0, the default specified using **define_dlur_defaults** is used. This parameter is ignored if the *dspu_services* parameter is not set to DLUR.

dlus_retry_limit
Retry count for contacting a DLUS. This parameter is used to specify the number of times Communications Server for Linux should retry if it fails to contact the DLUS on the first attempt.

Specify a value in the range 1–65,534, or specify 65,535 to indicate that Communications Server for Linux should retry indefinitely until it contacts the DLUS.

conventional_lu_compression
Specifies whether data compression is requested for LU 0–3 sessions on this link. This parameter is used only if this link carries LU 0–3 traffic; it does not apply to LU 6.2 sessions. Possible values are:

YES Data compression should be used for LU 0–3 sessions on this link if the host requests it.

NO Data compression should not be used for LU 0–3 sessions on this link.

branch_link_type
This parameter applies only if the local node is a Branch Network Node; it is not used if the local node is any other type.

If the parameter *adj_cp_type* is set to NETWORK_NODE, END_NODE, APPN_NODE, or BACK_LEVEL_LEN_NODE, this parameter defines whether the link is an uplink or a downlink. Possible values are:

UPLINK The link is an uplink.

DOWNLINK
The link is a downlink.

If *adj_cp_type* is set to NETWORK_NODE, this parameter must be set to UPLINK.

adj_brnn_cp_support
This parameter applies only if the local node is a Branch Network Node and the adjacent node is a network node (the parameter *adj_cp_type* is set to NETWORK_NODE, or it is set to APPN_NODE and the node type discovered during XID exchange is network node). It is not used if the local and remote nodes are any other type.

This parameter defines whether the adjacent node can be a Branch Network Node that is acting as a Network Node from the point of view of the local node. Possible values are:

ALLOWED
The adjacent node is allowed (but not required) to be a Branch Network Node.

REQUIRED
The adjacent node must be a Branch Network Node.

PROHIBITED
The adjacent node must not be a Branch Network Node.

If *adj_cp_type* is set to NETWORK_NODE and *auto_act_supp* is set to YES, this parameter must be set to REQUIRED or PROHIBITED.

initially_active
Specifies whether this LS is automatically started when the node is started. Possible values are:

YES The LS is automatically started when the node is started.

NO The LS is not automatically started; it must be manually started.

react_timer
Reactivation timer for reactivating a failed LS. If the *react_timer_retry* parameter is a nonzero value (to specify that Communications Server for Linux should retry activating the LS if it fails), this parameter specifies the time in seconds between retries. When the LS fails, or when an attempt to reactivate it fails, Communications Server for Linux waits for the specified time before retrying the activation. If *react_timer_retry* is 0 (zero), this parameter is ignored.

react_timer_retry
Retry count for reactivating a failed LS. This parameter is used to specify whether Communications Server for Linux attempts to reactivate the LS if it fails while in use (or if an attempt to start the LS fails).

Specify 0 (zero) to indicate that Communications Server for Linux should not attempt to reactivate the LS, or specify the number of retries to be made. A value of 65,535 indicates that Communications Server for Linux should retry indefinitely until the LS is activated.

Communications Server for Linux waits for the time specified by the *react_timer* parameter between successive retries. If the LS is not successfully reactivated by the end of the retry count, or if **stop_ls** is issued while Communications Server for Linux is retrying the activation, no further retries are made; the LS remains inactive unless **start_ls** is issued for it.

If the *auto_act_supp* parameter is set to YES, the *react_timer* and *react_timer_retry* parameters are ignored; if the link fails, Communications Server for Linux does not attempt to reactivate it until the user application that was using the session attempts to restart the session.

restart_on_normal_deact
Specifies whether Communications Server for Linux should attempt to reactivate the LS if it is deactivated normally by the remote system. Possible values are:

YES If the remote system deactivates the LS normally, Communications Server for Linux attempts to reactivate it, using the same retry timer and count values as for reactivating a failed LS (the *react_timer* and *react_timer_retry* parameters above).

NO If the remote system deactivates the LS normally, Communications Server for Linux does not attempt to reactivate it.

If the LS is a host link (specified by the *adj_cp_type* parameter), or is automatically started when the node is started (the *initially_active* parameter is set to YES), this parameter is ignored; Communications Server for Linux always attempts to reactivate the LS if it is deactivated normally by the remote system (unless *react_timer_retry* is zero).

xid_timer
Timeout required before an XID is retransmitted when trying to contact a remote station. The timer is specified in seconds. Higher values may be needed if the propagation delay to the remote station is large.

xid_timer_retry
Number of times transmission and retransmission of an XID is allowed. This count does not include the initial transmission; a value of 1 indicates "transmit once and then retry once." Higher values may be needed if the link to the remote station is unreliable or can become congested.

test_timeout
Timeout required before a TEST frame is retransmitted when trying to contact a remote station. The timer is specified in seconds. Higher values may be needed if the propagation delay to the remote station is large.

test_timer_retry
Number of times transmission and retransmission of a TEST frame is allowed. This count does not include the initial transmission; a value of 1 indicates "transmit once and then retry once." Higher values may be needed if the link to the remote station is unreliable or can become congested.

ack_timeout
Acknowledgment timeout—the time in milliseconds within which a response must be received for any I-frames sent to the adjacent link station.

p_bit_timeout
Poll bit timeout—the time in milliseconds within which a response must be received for any frames sent to the adjacent link station with the POLL bit set.

t2_timeout
The maximum time in milliseconds that the local station can wait before it must send a response to a received I-frame. A longer timeout enables the local station to respond to more than one I-frame with a single RR and therefore reduces acknowledgment traffic.

rej_timeout
Reject timeout—the time in seconds within which a response must be received for an REJ frame sent to the adjacent link station.

busy_state_timeout
The time in seconds that the local station waits for indication from the adjacent link station that a busy state of receive not ready (RNR) has cleared.

idle_timeout
Idle timeout. This parameter is used to send keep-alive (RR or RNR) frames to the remote station. The line is considered idle when nothing has been received in the specified time. The timer is specified in seconds.

max_retry
The maximum number of times that the local station retries when waiting for a response or for a busy state to clear.

dddlu_offline_supported
Specifies whether the local PU should send NMVT (power off) messages to the host. If the host system supports DDDLU (Dynamic Definition of Dependent LUs), Communications Server for Linux sends NMVT (power off) to the host when it has finished using a dynamically defined LU. This allows the host to save resources by removing the definition when it is no longer required.

This parameter is used only if this link is to a host (*solicit_sscp_sessions* is set to `YES` and *dspu_services* is not set to `NONE`).

Possible values are:

YES The local PU sends NMVT (power off) messages to the host.

NO The local PU does not send NMVT (power off) messages to the host.

If the host supports DDDLU but does not support the NMVT (power off) message, this parameter must be set to `NO`.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
`PARAMETER_CHECK`

secondary_rc
Possible values are:

CANT_MODIFY_PORT_NAME
The *ls_name* parameter matched the name of an existing LS, but the *port_name* parameter did not match the existing definition. You cannot modify the port name when changing the definition of an existing LS.

DEF_LINK_INVALID_SECURITY
The *security* parameter was not set to a valid value.

INVALID_AUTO_ACT_SUPP
The *auto_act_supp* parameter was not set to a valid value or was set to YES when *cp_cp_sess_support* was also set to YES.

INVALID_CP_NAME
The *adj_cp_name* parameter contained a character that was not valid, was not in the correct format, or was not specified when required.

INVALID_LIMITED_RESOURCE
The *limited_resource* parameter was not set to a valid value.

INVALID_LINK_NAME
The *ls_name* parameter contained a character that was not valid.

INVALID_NODE_TYPE
The *adj_cp_type* parameter was not set to a valid value.

INVALID_PORT_NAME
The *port_name* parameter did not match the name of any defined port.

INVALID_PU_NAME
The *pu_name* parameter did not match the name of any defined PU, or was set to a new value on an already-defined LS.

INVALID_DSPU_NAME
The *dspu_name* parameter did not match the name of any defined PU, or was set to a new value on an already-defined LS.

INVALID_DSPU_SERVICES
The *dspu_services* parameter was not set to a valid value or was set when not expected.

INVALID_SOLICIT_SSCP_SESS
The *solicit_sscp_sess* parameter was not set to a valid value.

INVALID_TARGET_PACING_CNT
The *target_pacing_count* parameter was not set to a valid value.

INVALID_DLUS_NAME
The *dlus_name* parameter contained a character that was not valid or was not in the correct format.

INVALID_BKUP_DLUS_NAME
The *bkup_dlus_name* parameter contained a character that was not valid or was not in the correct format.

HPR_NOT_SUPPORTED
A reserved parameter was set to a nonzero value.

INVALID_TG_NUMBER
The TG number supplied was not in the valid range.

MISSING_CP_NAME
A TG number was defined, but no CP name was supplied.

MISSING_CP_TYPE
A TG number was defined, but no CP type was supplied.

MISSING_TG_NUMBER
The link was defined to be automatically activated, but no TG number was supplied.

INVALID_BRANCH_LINK_TYPE
The *branch_link_type* parameter was not set to a valid value.

INVALID_BRNN_SUPPORT
The *adj_brnn_cp_support* parameter was not set to a valid value.

BRNN_SUPPORT_MISSING
The *adj_brnn_cp_support* parameter was set to ALLOWED; this value is not valid because the adjacent node is a Network Node and *auto_act_supp* is set to YES.

INVALID_UPLINK
The *branch_link_type* parameter was set to UPLINK, but the definition of an existing LS between the local and adjacent nodes specifies that it is a downlink. The branch link type must be the same for all LSs between the same two nodes.

INVALID_DOWNLINK
The *branch_link_type* parameter was set to DOWNLINK, but the definition of an existing LS between the local and adjacent nodes specifies that it is an uplink. The branch link type must be the same for all LSs between the same two nodes.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

DUPLICATE_DEST_ADDR
A link to the destination address specified by the combination of the *mac_address* and *lsap_address* parameters has already been defined.

INVALID_NUM_LS_SPECIFIED
The number of link stations specified was not valid.

LOCAL_CP_NAME
The value specified in the *adj_cp_name* parameter was the same as the local CP name.

LS_ACTIVE
The link station specified in the *ls_name* parameter is currently active.

PU_ALREADY_DEFINED
The PU specified in the *pu_name* parameter has already been defined.

DSPU_ALREADY_DEFINED
The downstream PU specified in the *dspu_name* parameter has already been defined.

DSPU_SERVICES_NOT_SUPPORTED
The *dspu_services* parameter was used to request a service that is not supported.

DUPLICATE_TG_NUMBER
The TG number specified in the *tg_number* parameter has already been defined.

TG_NUMBER_IN_USE
The TG number specified in the *tg_number* parameter is in use by another link station.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

Bit Ordering in MAC Addresses

Ethernet LANs use a different representation of MAC addresses from that used by Token Ring. The order of the bits in each byte of the address on Ethernet is the reverse of the order on Token Ring. The local and remote nodes are usually either on the same LAN or on LANs of the same type connected by a bridge; in both cases, the nodes will both use the same representation of the MAC address, and no conversion is required.

If the two nodes are on LANs of different types (one Ethernet, the other Token Ring) connected by a bridge, you will usually need to reverse the bit order of each byte of the address when specifying a remote MAC address. To reverse the bit order, take the following steps:

Reversing the Bit Order in a MAC Address

1. List the MAC address as six bytes, with each byte represented by two hexadecimal digits.
2. Swap the order of the two digits of each byte.
3. Convert each digit as shown in Table 3.

Table 3. Bit Conversion for MAC Addresses

0→0	8→1
1→8	9→9
2→4	A→5
3→C	B→D
4→2	C→3
5→A	D→B
6→6	E→7
7→E	F→F

Table 4 illustrates steps 1, 2, and 3:

Table 4. MAC Address Bit Conversion Example

List the MAC address	1A 2B 3C 4D 5E 6F
Swap the digit order	A1 B2 C3 D4 E5 F6
Convert each digit	58 D4 3C B2 7A F6 (the bit-reversed form of the original address)

define_tr_port, define_ethernet_port

The **define_tr_port** command is used to define a new Token Ring port or modify an existing port. Before issuing this command, you must define the DLC that this port uses.

The **define_ethernet_port** command is used to define a new Ethernet port or modify an existing port. Before issuing this command, you must define the DLC that this port uses. The parameters and defaults are the same as for **define_tr_port**, except where noted.

You can modify an existing port only if it is not started. You cannot change the DLC used by an existing port; the name specified in the *dlc_name* parameter when modifying an existing port must match the DLC that was specified on the initial definition of the port.

For more information about defining a port that accepts incoming calls, see "Incoming Calls" on page 232.

Supplied Parameters

Parameter name	Type	Length	Default
[define_tr_port], [define_ethernet_port]			
port_name	character	8	
description	character	31	(null string)
dlc_name	character	8	
port_number	decimal		1
max_rcv_btu_size	decimal		4105 (TR) 1033 (Ethernet)
tot_link_act_lim	decimal		64
inb_link_act_lim	decimal		0
out_link_act_lim	decimal		0
ls_role	constant		LS_NEG
implicit_dspu_services	constant		NONE
implicit_dspu_template	character	8	(null string)
implicit_ls_limit	decimal		
act_xid_exchange_limit	decimal		9
nonact_xid_exchange_limit	decimal		5
ls_xmit_rcv_cap	constant		LS_TWS
max_ifrm_rcvd	decimal		7
target_pacing_count	decimal		7
max_send_btu_size	decimal		4105 (TR) 1033 (Ethernet)
lsap_address	hex number		0x04
implicit_cp_cp_sess_support	constant		YES
implicit_limited_resource	constant		NO
implicit_hpr_support	constant		YES
implicit_link_lvl_error	constant		
implicit_deact_timer	decimal		30
implicit_uplink_to_en	constant		NO
effect_cap	decimal		16000000 (TR) 865075200 (Ethernet, Linux on System z) 157286400 (Ethernet, other Linux variants)
connect_cost	decimal		0
byte_cost	decimal		0
security	constant		SEC_NONSECURE
prop_delay	constant		PROP_DELAY_LAN
user_def_parm_1	decimal		0
user_def_parm_2	decimal		0
user_def_parm_3	decimal		0
initially_active	constant		YES
test_timeout	decimal		10
test_timer_retry	decimal		5
xid_timer	decimal		10
xid_timer_retry	decimal		5
ack_timeout	decimal		5000
p_bit_timeout	decimal		5000
t2_timeout	decimal		100
rej_timeout	decimal		10
busy_state_timeout	decimal		30
idle_timeout	decimal		30
max_retry	decimal		3
window_inc_threshold	decimal		1

Supplied parameters are:

port_name
: Name of the port to be defined. This name is a character string using any locally displayable characters.

description
: A text string describing the port. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_port** command.

dlc_name
Name of the associated DLC. This name is a character string using any locally displayable characters. The specified DLC must have already been defined.

port_number
The number of the port.

max_rcv_btu_size
Maximum BTU size that can be received. This value includes the length of the TH and RH (total 9 bytes) as well as the RU. Specify a value in the range `265–65535`.

tot_link_act_lim
Total link activation limit (the maximum number of links that can be active at any time using this port).

inb_link_act_lim
Inbound link activation limit (the number of links reserved for inbound activation). The sum of *inb_link_act_lim* and *out_link_act_lim* must not exceed *tot_link_act_lim*; the difference between *inb_link_act_lim* and *tot_link_act_lim* defines the maximum number of links that can be activated outbound at any time.

out_link_act_lim
Outbound link activation limit (the number of links reserved for outbound activation). The sum of *inb_link_act_lim* and *out_link_act_lim* must not exceed *tot_link_act_lim*; the difference between *out_link_act_lim* and *tot_link_act_lim* defines the maximum number of links that can be activated inbound at any time.

ls_role Link station role. Set this to `LS_NEG`.

implicit_dspu_services
Specifies the services that the local node will provide to the downstream PU across implicit links activated on this port. Possible values are:

DLUR Local node will provide DLUR services for the downstream PU (using the default DLUS configured through the **define_dlur_defaults** command).

PU_CONCENTRATION
Local node will provide SNA gateway for the downstream PU. It will also put in place definitions as specified by the DSPU template specified for the parameter *implicit_dspu_template*.

NONE Local node will provide no services for this downstream PU.

implicit_dspu_template
Specifies the DSPU template, defined on the **define_dspu_template** command. This template is used for definitions if the local node is to provide SNA gateway for an implicit link activated on this port. If the template specified does not exist or is already at its instance limit when the link is activated, activation will fail. This template name is an 8-byte string in a locally displayable character set.

If the *implicit_dspu_services* parameter is not set to `PU_CONCENTRATION`, the *implicit_dspu_template* parameter is reserved.

implicit_ls_limit
Specifies the maximum number of implicit link stations which can be active on this port simultaneously, including dynamic links and links

activated for Discovery. Specify a value in the range 1–65,534 or specify 0 (zero) to indicate no limit. A value of NO_IMPLICIT_LINKS indicates that no implicit links are allowed.

act_xid_exchange_limit
Activation XID exchange limit. Specify a value in the range 0–65,535.

nonact_xid_exchange_limit
Nonactivation XID exchange limit. Specify a value in the range 0–65,535.

ls_xmit_rcv_cap
Specifies the link station transmit/receive capability. Possible values are:

LS_TWS Two-way simultaneous

LS_TWA Two-way alternating

max_ifrm_rcvd
Maximum number of I-frames that can be received by the local link stations before an acknowledgment is sent. Specify in the range 1–127.

target_pacing_count
Indicates the desired pacing window size. Specify a value in the range 1–32,767.

max_send_btu_size
Maximum BTU size that can be sent from this port. This value is used to negotiate the maximum BTU size that a pair of link stations can use to communicate with each other. The value includes the length of the TH and RH (total 9 bytes), as well as the RU. Specify a value in the range 265–65535.

lsap_address
Local SAP address of the port. Specify a multiple of 0x04 in the range 0x04–0xEC. The value must be specified as two hexadecimal digits preceded by 0x.

implicit_cp_cp_sess_support
Specifies whether CP-CP sessions are allowed for implicit link stations using this port. Possible values are:

YES CP-CP sessions are allowed for implicit link stations.

NO CP-CP sessions are not allowed for implicit link stations.

implicit_limited_resource
Specifies whether implicit link stations off this port should be defined as limited resources. Possible values are:

NO Implicit links are not limited resources, and are not automatically deactivated.

NO_SESSIONS
Implicit links are limited resources, and are automatically deactivated when no active sessions are using them.

INACTIVITY
Implicit links are limited resources, and are automatically deactivated when no active sessions are using them or when no data has flowed for the time period specified by the *implicit_deact_timer* parameter.

- If no SSCP-PU session is active across the link, the node deactivates the link immediately.

- If an SSCP-PU session is active but no traffic has flowed for the specified time period, the node sends REQDISCONT(normal) to the host. The host is then responsible for deactivating all LUs and the PU, at which time the local node will deactivate the link. However, the host may not deactivate LUs with active PLU-SLU sessions; in this case, the link remains active until all these sessions are deactivated (for example by the user logging out). This behavior can be changed by using options in the *ptf* parameter of the **start_node** command.

implicit_hpr_support
Specifies whether High Performance Routing (HPR) is supported on implicit links. Possible values are:

YES HPR is supported on implicit links.

NO HPR is not supported on implicit links.

implicit_link_lvl_error
Specifies whether HPR traffic should be sent on implicit links using link-level error recovery. This parameter is ignored if *implicit_hpr_support* is set to `NO`. Possible values are:

YES HPR traffic should be sent on implicit links using link-level error recovery.

NO HPR traffic should not be sent on implicit links using link-level error recovery.

implicit_deact_timer
Implicit limited resource link deactivation timer, in seconds.

If *implicit_hpr_support* is set to `YES` and *implicit_limited_resource* is set to `NO_SESSIONS`, an implicit link is automatically deactivated if no data flows on it for the time specified by this parameter and no sessions are using the link.

If *implicit_limited_resource* is set to `INACTIVITY`, an implicit link using this port is automatically deactivated if no data flows on it for the time specified by this parameter.

The minimum value is `5`; values in the range `1–4` will be interpreted as `5`. The value `0` (zero) indicates no timeout (the link is not deactivated, as though *implicit_limited_resource* were set to `NO`). This parameter is reserved if *implicit_limited_resource* is set to `NO`.

implicit_uplink_to_en
This parameter applies only if the local node is a Branch Network Node; it is reserved if the local node is any other type.

If the adjacent node is an end node, this parameter specifies whether implicit link stations off this port are uplink or downlink. This parameter is ignored if there are existing links to the same adjacent node, because in this case the existing links are used to determine the link type. Possible values are:

YES Implicit links to an End Node are uplinks.

NO Implicit links to an End Node are downlinks.

effect_cap **through** *user_def_parm_3*
Default TG characteristics used for implicit link stations using this port and as the default TG characteristics for defined link stations that do not have

explicitly defined TG characteristics. The TG characteristics parameters are ignored if the LS is to a downstream PU.

For more information about these parameters, see "define_tr_ls, define_ethernet_ls" on page 209.

initially_active
Specifies whether this port is automatically started when the node is started. Possible values are:

YES The port is automatically started when the node is started.

NO The port is automatically started only if an LS that uses the port is defined as initially active; otherwise, it must be manually started.

test_timeout **through** *max_retry*
For information about these parameters, see "define_tr_ls, define_ethernet_ls" on page 209. When the LS name is not initially known, the values specified on **define_tr_port** / **define_ethernet_port**are used as defaults for processing incoming calls.

window_inc_threshold
The number of I-frames that must be acknowledged successfully before the working window size is incremented. This value is used by the dynamic window algorithm to increase the window size after it has been reduced following an error condition.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_PORT_NAME
The value specified in the *port_name* parameter was not valid.

INVALID_DLC_NAME
The specified *dlc_name* did not match any defined DLC.

INVALID_PORT_TYPE
The *port_type* parameter was not set to a valid value.

INVALID_BTU_SIZE
The *max_rcv_btu_size* parameter was not set to a valid value.

INVALID_LS_ROLE
The *ls_role* parameter was not set to a valid value.

INVALID_LINK_ACTIVE_LIMIT
One of the activation limit parameters, *inb_link_act_lim*, *out_link_act_lim*, or *tot_link_act_lim*, was not set to a valid value.

INVALID_MAX_IFRM_RCVD
The *max_ifrm_rcvd* parameter was not set to a valid value.

HPR_NOT_SUPPORTED
A reserved parameter was set to a nonzero value.

DLUR_NOT_SUPPORTED
The *implicit_dspu_services* parameter was used to request a service that is not supported.

PU_CONC_NOT_SUPPORTED
The *implicit_dspu_services* parameter was used to request a service that is not supported.

INVALID_IMPLICIT_UPLINK
The *implicit_uplink_to_en* parameter was not set to a valid value.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

PORT_ACTIVE
The specified port cannot be modified because it is currently active.

DUPLICATE_PORT_NUMBER
A port with the number specified in the *port_number* parameter has already been defined.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

Incoming Calls

If you are configuring a port that accepts incoming calls (as defined by the *tot_link_act_lim*, *inb_link_act_lim*, and *out_link_act_lim* parameters), there is generally no need to define an LS to use for these calls; Communications Server for Linux will dynamically define an LS when the incoming call is received. However, if the incoming calls are from a host computer that supports dependent LUs or from a downstream computer using SNA gateway, you need to explicitly define an LS because the LS definition includes the name of the PU associated with the dependent LUs or the name of the downstream PU.

When an incoming call arrives at the port, Communications Server for Linux checks the MAC and SAP addresses specified on the call against the addresses specified for link stations defined on the port (if any) to determine if an LS has already been defined for the call. If the MAC / SAP address pair does not match the MAC / SAP address pair specified on any of these link stations, an LS is dynamically defined. To ensure that the explicit LS definition (including the

required PU name) is used, ensure that both the MAC and SAP addresses defined for this LS match the addresses that are supplied by the host or the downstream computer on the incoming call.

define_userid_password

The **define_userid_password** command defines a user ID / password pair for use with APPC and CPI-C conversation security, or adds profiles for a defined user ID and password.

Supplied Parameters

Parameter name [define_userid_password]	Type	Length	Default
define_type	constant		ADD_USER
user_id	character	10	
description	character	31	(null string)
password	character	10	
profile	character	10	(null string)

(Up to ten *profile* parameters can be specified.)

Supplied parameters are:

define_type
Specifies how this command is to be used. Possible values are:

ADD_USER
Add a new user, or change the password for an existing user.

ADD_PROFILES
Add profiles to an existing user id/password record.

user_id User identifier. This name is a type-AE character string. Some CPI-C implementations have a maximum user ID length of eight characters. If you specify a user ID of nine or ten characters, CPI-C applications running on other systems may not be able to access applications on the Communications Server for Linux system using this user ID and password.

description
A text string describing the user ID and password. Communications Server for Linux uses this string for information only. It is stored in the node's configuration file and returned on the **query_userid_password** command.

password
User's password. This password is a type-AE character string. Some CPI-C implementations have a maximum password length of eight characters. If you specify a password of nine or ten characters, CPI-C applications running on other systems may not be able to access applications on the Communications Server for Linux system using this user ID and password.

When you type in this parameter on the command line, the value you type in is immediately replaced by the encrypted version of the password. Therefore, the value you supply for the *password* parameter is never displayed on the command line.

profile Profile associated with user. Each profile is a type-AE character string.

If a remote TP uses the user ID and password specified for this command when attaching to the local TP, the profile specified on the Attach (if any) must match one of the profile names defined for this command. Consult the System Administrator running the remote TP to determine if profiles

are used. For each profile used, specify the profile name as one of the *profile* parameters on this command. In most cases, profile names are not used, therefore you do not need to specify them on this command.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_PASSWORD
The *password* parameter contained a character that was not valid.

INVALID_PROFILE
One or more of the specified *profile* values were not valid.

INVALID_USERID
The *user_id* parameter contained a character that was not valid.

NO_PROFILES
The command was used to add profiles to an existing user, but no profiles were specified.

UNKNOWN_USER
The command was used to add profiles to an existing user, but the *user_id* parameter did not match an existing user ID.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

delete_adjacent_len_node

The **delete_adjacent_len_node** command deletes entries in the node directory database for an adjacent LEN node and its associated LUs, or removes LU entries for the LEN node without removing the LEN node itself. It is equivalent to issuing a series of **delete_directory_entry** commands for the LEN node and its associated LUs.

Supplied Parameters

Parameter name [delete_adjacent_len_node]	Type	Length	Default
cp_name	character	17	
lu_name	character	8	
wildcard_lus	constant		NO

(Up to ten *lu_name* parameters can be specified.)

Supplied parameters are:

cp_name
The fully qualified name of the CP in the adjacent node. Specify 3–17 type-A characters consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character CP name.

lu_name
The name of each LU to be deleted. Each name is an 8-byte type-A character string. To delete the entire LEN node definition, do not specify any LU names.

You can specify a "wildcard" LU name to match multiple LU names, by specifying only the initial characters of the name. For example, the wildcard LU name APPN.LU will match APPN.LUNAME or APPN.LU01 (but will not match APPN.NAMELU). However, all the LU names specified on a single command must be of the same type (wildcard or explicit), as defined by the *wildcard_lus* parameter. To remove both types of LU names from the same LEN node, use multiple **delete_adjacent_len_node** commands.

wildcard_lus
Indicates whether the specified LU names are wildcard entries or explicit LU names. Possible values are:

YES The specified LU names are wildcard entries.

NO The specified LU names are explicit entries.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_CP_NAME
The *cp_name* parameter contained a character that was not valid.

INVALID_LU_NAME
One or more of the specified LU names contained a character that was not valid.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

INVALID_CP_NAME
The specified CP name does not match the name of a defined directory entry.

INVALID_LU_NAME
One or more of the specified LU names does not match any defined LU name.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

delete_backup

The **delete_backup** command deletes a server from the list of servers in the **sna.net** file; this server can no longer act as the master configuration file server.

You can use this command to delete any server in the list, including the master server, whether or not the SNA software is running on the server you are deleting. The only restriction is that the list must always contain at least one server on which the SNA software is running (so that this server can take over as the master server). You cannot delete a server if it is the only server in the list or if it is the only server listed on which the SNA software is running.

This command must be issued without specifying a node name.

Supplied Parameters

Parameter name	Type	Length
[delete_backup]		
backup_name	character	128

Supplied parameter is:

backup_name
The name of the server to be deleted from the list of backup servers.

If the server name includes a **.** (period) character, Communications Server for Linux assumes that it is a fully-qualified name; otherwise it performs a DNS lookup to determine the server name.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc

STATE_CHECK

secondary_rc

Possible values are:

RECORD_NOT_FOUND

The server name specified in the *backup_name* parameter is not listed in the file.

CANT_DELETE_LAST_BACKUP

The server name cannot be deleted from the list because it is the only server listed on which the SNA software is running (the only server that can currently act as the master server). Before attempting to delete the server, either start the SNA software on one or more of the other servers listed, or add one or more new backup servers (using **add_backup**) and ensure that the SNA software is started on these servers.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

delete_cn

The **delete_cn** command deletes a connection network or deletes selected ports from a connection network.

This command is valid only at a network node or an end node; it is not valid at a low-entry networking (LEN) node.

Supplied Parameters

Parameter name	Type	Length
[delete_cn]		
fqcn_name	character	17
port_name	character	8

(One or more *port_name* entries can be included.)

Supplied parameters are:

fqcn_name

Specifies the fully qualified name of the connection network. Specify 3–17 type-A characters consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character connection network name.

port_name
If you are deleting ports without deleting the connection network, specify the names of the ports to be deleted. Each port name is a string of up to eight characters. To delete the connection network, do not specify any port names.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_CN_NAME
The *fqcn_name* parameter was not set to a valid CN name.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Function Not Supported

If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameters:

primary_rc

FUNCTION_NOT_SUPPORTED
The local node is a LEN node. This command is valid only at a network node or an end node.

secondary_rc
(This parameter is not used.)

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

delete_cos

The **delete_cos** command deletes a class of service (COS) entry. Only locally-defined COSs can be deleted; the default COSs defined by SNA cannot be deleted.

If the node supports mode-to-COS mapping (as defined by the *mode_to_cos_map_supp* parameter on the **define_node** command) and the

configuration includes modes that are mapped to the COS you are deleting, Communications Server for Linux will remap these modes to the default COS (specified by a **define_mode** command with no mode name) or to the SNA-defined COS #CONNECT if no default COS is specified.

Supplied Parameters

```
Parameter name          Type          Length
[delete_cos]
cos_name                character     8
```

Supplied parameter is:

cos_name
Specifies the class of service name to be deleted. This name is type-A character string starting with a letter.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

COS_NAME_NOT_DEFD
The supplied name is not the name of a COS defined on the Communications Server for Linux node.

SNA_DEFD_COS_CANT_BE_DELETED
The supplied name is the name of an SNA-defined COS, which cannot be deleted.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

delete_cpic_side_info

The **delete_cpic_side_info** command deletes a CPI-C side information entry.

Because CPI-C side information entries are defined as domain resources, this command is not associated with a particular node.

Supplied Parameters

Parameter name	Type	Length
[delete_cpic_side_info]		
sym_dest_name	character	8

Supplied parameter is:

sym_dest_name
Symbolic destination name that identifies the side information entry. Specify any locally displayable character.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

INVALID_SYM_DEST_NAME
The *sym_dest_name* parameter was not the name of a defined CPI-C side information entry.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

delete_directory_entry

The **delete_directory_entry** command deletes an entry in the Network Directory. You cannot delete the entry for an end node CP from the directory of its network node server.

When the entry for a parent resource is deleted, then all entries for child resources associated with it are also deleted. For example, when you delete the entry for a network node that is the parent of an end node, then the entries for the end node and all LUs associated with both nodes (including wildcard LU entries) are deleted as well as the entry for the network node.

Supplied Parameters

Parameter name	Type	Length	Default
[delete_directory_entry]			
resource_name	character	17	
resource_type	constant		LU_RESOURCE

Supplied parameters are:

resource_name
Fully qualified name of the resource to be deleted. Specify 3–17 type-A characters consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character resource name.

resource_type
Specifies the type of the resource to be deleted. Possible values are:

ENCP_RESOURCE
End node (EN) or low-entry networking (LEN) node

NNCP_RESOURCE
Network node (NN)

LU_RESOURCE
Logical unit (LU)

WILDCARD_LU_RESOURCE
Wildcard LU name

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_FQ_LU_NAME
The *resource_name* parameter was not the name of a defined LU.

INVALID_RESOURCE_TYPE
The *resource_type* parameter was not set to a valid value.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc

CANT_DELETE_ADJ_ENDNODE
The specified entry is for an end node, and the node to which this command was issued is its network node server. You cannot delete this end node entry.

Other Conditions
Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

delete_dlc

The *delete_dlc* command deletes a DLC. The command also deletes the following:

- All ports, link stations, and connection network TGs associated with the DLC
- All PUs associated with link stations on the DLC, all LUs owned by these PUs, and all LU-LU passwords associated with these LUs

Supplied Parameters

```
Parameter name      Type        Length
[delete_dlc]
dlc_name            character   8
```

Supplied parameter is:

dlc_name
Name of DLC to be deleted.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

INVALID_DLC_NAME
The specified *dlc_name* did not match any defined DLC.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc

DLC_ACTIVE
The DLC cannot be deleted because it is currently active. Use **stop_dlc** to stop the DLC before attempting to delete it.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

delete_downstream_lu

The **delete_downstream_lu** command deletes a downstream LU.

Supplied Parameters

```
Parameter name          Type          Length
[delete_downstream_lu]
dslu_name               character     8
```

The supplied parameter is:

dslu_name
Name of the downstream LU to be deleted. This name is a type-A character string starting with a letter.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

INVALID_LU_NAME
The *dslu_name* parameter contained a character that was not valid.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

INVALID_LU_NAME
The *dslu_name* parameter did not match any defined downstream LU name.

DSLU_ACTIVE
The LU cannot be deleted because it is currently active.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

delete_downstream_lu_range

The **delete_downstream_lu_range** command deletes a range of downstream LUs. The supplied parameters include a base name for the LUs and the range of NAU addresses. The LU base name and NAU addresses are combined to determine the range of LUs to delete. For example, a base name of LUNME combined with a NAU range of 11–14 deletes the following LUs: LUNME011, LUNME012, LUNME013, and LUNME014.

All LUs with names in the specified range are deleted; Communications Server for Linux does not return an error if one or more names in the range do not exist.

Supplied Parameters

```
Parameter name           Type           Length      Default
[delete_downstream_lu_range]
dslu_base_name           character      5
min_nau                  decimal                    1
max_nau                  decimal                    1
```

Supplied parameters are:

dslu_base_name
Base name for the names of the LUs to be deleted. This name is a type-A character string of 1–5 characters starting with a letter. Communications Server for Linux appends the 3-digit decimal value of each NAU address to this name to determine which LUs to delete.

min_nau
NAU address of the first LU to be deleted, in the range 1–255.

max_nau
NAU address of the last LU to be deleted, in the range 1–255.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_NAU_ADDRESS
The *min_nau* or *max_nau* parameter value was not valid.

INVALID_LU_NAME
The *dslu_base_name* parameter contained a character that was not valid.

State Check
If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

INVALID_LU_NAME
There were no LUs defined with names in the specified range.

DSLU_ACTIVE
One or more of the LUs in the range cannot be deleted because it is currently active.

Other Conditions
Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

delete_dspu_template

The **delete_dspu_template** command deletes a specific DSPU template previously defined on a **define_dspu_template** command, or deletes one or more downstream LU (DSLU) templates from a DSPU template.

Supplied Parameters

Parameter name	Type	Length	Default
[delete_dspu_template]			
template_name	character	8	
{dslu_template}			
min_nau	decimal		
max_nau	decimal		
host_lu	character	8	
allow_timeout	constant		NO
delayed_logon	constant		NO

Supplied parameters are:

template_name
Name of the DSPU template to be deleted, or the DSPU template containing the DSLU templates to be deleted. Specify 1–8 locally displayable characters.

To delete the entire DSPU template, do not specify any `dslu_template` subrecords. To delete one or more DSLU templates but leave the DSPU template configured, specify a `dslu_template` subrecord for each DSLU template to be deleted. The subrecord `dslu_template` contains the following parameters:

min_nau
Minimum NAU address in the range of DSLU templates to be deleted. Specify a value in the range 1–255.

max_nau
Maximum NAU address in the range of DSLU templates to be deleted. Specify a value in the range 1–255.

allow_timeout
Specifies whether Communications Server for Linux is allowed to timeout host LUs used by this downstream LU if the session is left inactive for the timeout period specified on the host LU definition. Possible values are:

YES Communications Server for Linux is allowed to timeout host LUs used by this downstream LU.

NO Communications Server for Linux is not allowed to timeout host LUs used by this downstream LU.

delayed_logon
Specifies whether Communications Server for Linux delays connecting the downstream LU to the host LU until the first data is received from the downstream LU. Instead, a simulated logon screen is sent to the downstream LU. Possible values are:

YES Communications Server for Linux delays connecting the downstream LU to the host LU until the first data is received from the downstream LU.

NO Communications Server for Linux does not delay connecting the downstream LU to the host LU until the first data is received from the downstream LU.

host_lu Name of the host LU or host LU pool onto which all the downstream LUs within the range will be mapped.

Returned Parameters

If the command executes successfully, the following parameter is returned:

primary_rc
OK

secondary_rc
(This parameter is not used.)

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_TEMPLATE_NAME
The template specified by the *template_name* parameter was not valid.

State Check
No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions
Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

delete_focal_point

The **delete_focal_point** command deletes the definition of a focal point for a specified MS category (either the main focal point for that category or a backup focal point). If the defined focal point application is active and acting as the current focal point for the specified MS category, Communications Server for Linux sends an MS_CAPABILITIES message to the focal point to revoke it so that it no longer acts as the focal point.

Supplied Parameters

Parameter name	Type	Length
[delete_focal_point]		
ms_category	character	8
type	constant	

Supplied parameters are:

ms_category
Management Services category. This parameter is one of the category names specified in *Systems Network Architecture: Management Services*, or a user-defined category. A user-defined category name is a type-1134 string.

type Specifies the type of focal point to be deleted. Possible values are:

ACTIVE The currently active focal point (any type) is revoked.

IMPLICIT
The implicit definition (defined using **define_focal_point** with *backup* set to NO) is deleted. If this focal point is currently active, it is revoked.

BACKUP The backup definition (defined using **define_focal_point** with *backup* set to YES) is deleted. If this focal point is currently active, it is revoked.

Returned Parameters
No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes
If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_CATEGORY_NAME
The *ms_category* parameter contained a character that was not valid.

INVALID_TYPE
The *type* parameter was not set to a valid value.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Function Not Supported

If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameters:

primary_rc

FUNCTION_NOT_SUPPORTED
The local node does not support MS network management functions; support is defined by the *mds_supported* parameter in the node definition.

secondary_rc
(This parameter is not used.)

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

delete_internal_pu

The **delete_internal_pu** command deletes a DLUR-served local PU that is served by DLUR. The PU can be deleted only if it does not have an active SSCP-PU session.

Supplied Parameters

Parameter name	Type	Length
[delete_internal_pu]		
pu_name	character	8

The supplied parameter is:

pu_name
Name of the internal PU to be deleted. This name is a type-A character string starting with a letter.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

INVALID_PU_NAME
The *pu_name* parameter was not the name of a defined internal PU.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

PU_NOT_RESET
The PU cannot be deleted because it still has an active PU-SSCP session.

INVALID_PU_TYPE
The specified PU is a remote PU, not an internal PU.

Function Not Supported

If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameters:

primary_rc

FUNCTION_NOT_SUPPORTED
The node does not support DLUR; support is defined by the *dlur_support* parameter on the **define_node** command.

secondary_rc
(This parameter is not used.)

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

delete_local_lu

The **delete_local_lu** command deletes a local LU and also deletes any LU-LU passwords associated with the LU.

Supplied Parameters

```
Parameter name          Type          Length
[delete_local_lu]
lu_name                 character     8
```

Supplied parameter is:

lu_name
Name of the local LU to be deleted. This name is a type-A character string starting with a letter.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

CANT_DELETE_CP_LU
The LU name associated with the CP was supplied; this LU cannot be deleted.

INVALID_LU_NAME
The supplied LU name is not the name of a local LU defined on the Communications Server for Linux system.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

delete_ls

The **delete_ls** command deletes a defined link station (LS). This command also deletes the PU associated with the LS, all LUs owned by this PU, and all LU-LU passwords associated with these LUs. The LS cannot be deleted if it is active.

Supplied Parameters

```
Parameter name          Type          Length
[delete_ls]
ls_name                 character     8
```

Supplied parameter is:

ls_name
Name of the link station to be deleted.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

INVALID_LINK_NAME
The supplied LS name contained a character that was not valid.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

LS_ACTIVE
The LS cannot be deleted because it is currently active.

INVALID_LINK_NAME
The supplied LS name is not the name of an LS defined on the Communications Server for Linux system.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

delete_ls_routing

The **delete_ls_routing** command deletes the association of a partner LU to a link station that was previously defined using the **define_ls_routing** command.

Supplied Parameters

Parameter name	Type	Length	Default
[delete_ls_routing]			
lu_name	character	8	
fq_partner_lu	character	17	
wildcard_fqplu	constant		NO

Supplied parameters are:

lu_name
Name of the local LU that communicated with the partner LU (specified by the *fq_partner_lu* parameter). Specify 1–8 locally displayable characters.

fq_partner_lu
Fully qualified name of the partner LU to be removed from the local LU's LS routing data. Specify 3–17 locally displayable characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character partner LU name.

To delete a wildcard entry, specify the same wildcard LU name that you used to define the entry. You cannot use wildcards to delete more than one explicitly-defined entry.

wildcard_fqplu
Wildcard partner LU flag indicating whether the *fq_partner_lu* parameter contains a full or partial wildcard. This flag is used to delete a wildcard entry; you cannot use wildcards to delete more than one explicitly-defined entry. Possible values are:

YES The *fq_partner_lu* parameter contains a wildcard entry.

NO The *fq_partner_lu* parameter does not contain a wildcard entry.

Returned Parameters

If the command executes successfully, the following parameter is returned:

primary_rc
OK

secondary_rc
(This parameter is not used.)

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_LOCAL_LU
The *lu_name* parameter contained a character that was not valid.

INVALID_PARTNER_LU
The *fq_partner_lu* parameter contained a character that was not valid.

INVALID_WILDCARD_NAME
The *wildcard_fqplu* parameter was set to YES, but the *fq_partner_lu* parameter was not a valid wildcard name.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

INVALID_LOCAL_LU
The *lu_name* parameter did not match an existing LS routing record.

INVALID_PARTNER_LU
The *fq_partner_lu* parameter did not match an existing LS routing record for the specified local LU.

INVALID_WILDCARD_NAME
The *wildcard_fqplu* parameter was set to YES, but no matching entry was found.

INVALID_RESOURCE_NAME
No LS routing entry that matched the supplied parameters was found.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

delete_lu_0_to_3

The **delete_lu_0_to_3** command is used to delete an LU used for 3270 emulation or LUA (an LU of type 0, 1, 2, or 3).

Supplied Parameters

Parameter name	Type	Length
[delete_lu_0_to_3]		
lu_name	character	8

Supplied parameter is:

lu_name
Name of the local LU to be deleted. This name is a type-A character string starting with a letter.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

INVALID_LU_NAME
The supplied LU name contained a character that was not valid.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc

INVALID_LU_NAME
The supplied LU name is not the name of an LU defined on the Communications Server for Linux node.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

delete_lu_0_to_3_range

The **delete_lu_0_to_3_range** command is used to delete a range of LUs used for 3270 emulationor LUA (LUs of type 0, 1, 2, or 3).

The supplied parameters include a base name for the LUs and the range of NAU addresses. The LU base name and NAU addresses are combined to determine the range of LUs to delete. For example, a base name of LUNME combined with a NAU range of 11–14 deletes the following LUs: LUNME011, LUNME012, LUNME013, and LUNME014.

All LUs with names in the specified range are deleted; Communications Server for Linux does not return an error if one or more names in the range do not exist.

Supplied Parameters

Parameter name [delete_lu_0_to_3_range]	Type	Length	Default
base_name	character	6	
name_attributes	constant		NONE
base_number	decimal		0
min_nau	decimal		1
max_nau	decimal		1

Supplied parameters are:

base_name
Base name for the names of the LUs. This name is a type-A character string of 1–5 characters starting with a letter. (However, if you specified USE_HEX_IN_NAME for the *name_attributes* parameter on the **define_lu_0_to_3_range** command, the base name can be 6 characters

long.) Communications Server for Linux determines the names of the LUs to be deleted by appending the 3-digit decimal value of each NAU address to this name.

name_attributes
Specifies the attributes of the LU names that are to be deleted.

Possible values are:

NONE LU names have numbers that correspond to the NAU numbers. The numbers are specified in decimal and the *base_name* parameter can contain only 5 characters.

USE_BASE_NUMBER
Start deleting the LUs in the range from the value specified in the *base_number* parameter.

USE_HEX_IN_NAME
The extension to the LU name is in hex rather than decimal. The *base_name* parameter can contain 6 characters if this value is specified.

base_number
If USE_BASE_NUMBER is specified in the *name_attributes* parameter, specify a number from which to start deleting the LUs in the range. This value will be used instead of the value of the *min_nau* parameter.

min_nau
NAU address of the first LU, in the range 1–255.

max_nau
NAU address of the last LU, in the range 1–255.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_NAU_ADDRESS
The value specified in the *min_nau* or *max_nau* parameter was not valid.

INVALID_LU_NAME
The *base_name* parameter contained a character that was not valid.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc

STATE_CHECK

secondary_rc

INVALID_LU_NAME

There were no LUs defined with names in the specified range.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

delete_lu_lu_password

The **delete_lu_lu_password** command deletes an LU-LU password associated with a local LU. LU-LU passwords are deleted automatically when the local LU is deleted; use this command only if you need to delete the password but leave the LU configured.

Supplied Parameters

Parameter name [delete_lu_lu_password]	Type	Length	Default
lu_name	character	8	(null string)
lu_alias	character	8	(null string)
fqplu_name	character	17	

Supplied parameters are:

lu_name

LU name of the local LU. This name is a type-A character string. To indicate that the LU is defined by its LU alias instead of its LU name, do not specify this parameter.

lu_alias

LU alias of the local LU. This is a character string using any locally displayable characters. This parameter is used only if *lu_name* is not specified.

To indicate the LU associated with the CP (the default LU), do not specify either *lu_name* or *lu_alias*.

fqplu_name

Fully qualified name of the partner LU. Specify 3–17 type-A characters consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character partner LU name.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_PLU_NAME
The *fqplu_name* parameter value was not valid.

INVALID_LU_NAME
The *lu_name* parameter value was not valid.

INVALID_LU_ALIAS
The *lu_alias* parameter value was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

delete_lu_pool

The **delete_lu_pool** command is used to do one of the following:

- Remove one or more LUs from a pool.
- Remove all LUs from a pool and delete the pool.

This command does not delete the LUs that have been removed from the pool; they remain defined but are not associated with any pool.

Supplied Parameters

Parameter name	Type	Length	Default
[delete_lu_pool]			
pool_name	character	8	
lu_name	character	8	

(Up to ten *lu_name* parameters can be specified.)

Supplied parameters are:

pool_name
Name of the LU pool to be deleted or name of the LU pool from which LUs are to be removed. This name is an 8–byte type-A character string.

lu_name
To remove one or more LUs from the pool without deleting the pool, specify the names of the LUs to be removed. Each name is a type-A character string starting with a letter.

To remove all LUs from the pool and delete the pool, do not specify any LU names.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_POOL_NAME
The supplied pool name was not valid.

INVALID_LU_NAME
One or more of the specified LU names did not match the name of an LU in the pool.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

delete_lu62_timeout

The **delete_lu62_timeout** command deletes a definition of an LU type 6.2 session timeout that was defined previously with a **define_lu62_timeout** command.

Supplied Parameters

Parameter name [delete_lu62_timeout]	Type	Length	Default
resource_type	constant		GLOBAL_TIMEOUT
resource_name	character	17	(null string)

Supplied parameters are:

resource_type
Specifies the type of timeout being deleted. Possible values are:

GLOBAL_TIMEOUT
Delete timeouts that apply to all LU 6.2 sessions for the local node.

LOCAL_LU_TIMEOUT
Delete timeouts that apply to all LU 6.2 sessions for the local LU specified in the *resource_name* parameter.

PARTNER_LU_TIMEOUT
Delete timeouts that apply to all LU 6.2 sessions to the partner LU specified in the *resource_name* parameter.

MODE_TIMEOUT
Delete timeouts that apply to all LU 6.2 sessions on the mode specified in the *resource_name* parameter.

resource_name
Name of the resource whose timeout is being deleted. This value can be one of the following:

- If *resource_type* is set to GLOBAL_TIMEOUT, do not specify this parameter.
- If *resource_type* is set to LOCAL_LU_TIMEOUT, specify 1–8 type-A characters as a local LU name.
- If *resource_type* is set to PARTNER_LU_TIMEOUT, specify the fully qualified name of the partner LU as follows: 3–17 type-A characters consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character partner LU name.
- If *resource_type* is set to MODE_TIMEOUT, specify 1–8 type-A characters as a mode name.

Returned Parameters

If the command executes successfully, the following parameter is returned:

primary_rc
OK

secondary_rc
(This parameter is not used.)

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_RESOURCE_TYPE
The value specified in the *resource_type* parameter was not valid.

INVALID_LU_NAME
The LU name specified in the *resource_name* parameter was not valid.

INVALID_PARTNER_LU
The partner LU name specified in the *resource_name* parameter was not valid.

INVALID_MODE_NAME
The mode name specified in the *resource_name* parameter was not valid.

GLOBAL_TIMEOUT_NOT_DEFINED
The value GLOBAL_TIMEOUT was specified for the *resource_type* parameter but there is no defined global timeout.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

delete_mode

The **delete_mode** command deletes the definition of a mode. You cannot delete SNA-defined modes such as SNASVCMG and CPSVCMG.

Supplied Parameters

Parameter name	Type	Length
[delete_mode]		
mode_name	character	8

Supplied parameter is:

mode_name
Name of the mode whose definition is to be deleted. This name is a type-A character string starting with a letter.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

CP_OR_SNA_SVCMG_UNDELETABLE
The specified mode name is one of the SNA-defined mode names and cannot be deleted.

MODE_NAME_NOT_DEFD
The specified mode name is not the name of a mode defined on the Communications Server for Linux system.

DEL_MODE_DEFAULT_SPCD
The specified mode was defined as the default mode using the **define_defaults** command, so it cannot be deleted.

MODE_UNDELETABLE
The specified mode name is one of the SNA-defined mode names and cannot be deleted.

State Check
No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions
Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

delete_partner_lu

The **delete_partner_lu** command deletes a partner LU definition.

Supplied Parameters

Parameter name	Type	Length
[delete_partner_lu]		
fqplu_name	character	17

Supplied parameter is:

fqplu_name
Fully qualified name of the partner LU. Specify 3–17 type-A characters consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character partner LU name.

Returned Parameters
No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes
If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check
If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

INVALID_PLU_NAME
The supplied *fqplu_name* parameter did not match any defined partner LU name.

State Check
No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

delete_port

The **delete_port** command deletes a port. This command also deletes the following:

- All link stations and connection network TGs associated with the port
- All PUs associated with link stations on the port and all LUs owned by these PUs

The port must be inactive when the command is issued.

Supplied Parameters

```
Parameter name          Type          Length
[delete_port]
port_name               character     8
```

Supplied parameter is:

port_name
Name of the port to be deleted.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

INVALID_PORT_NAME
The specified port name was not the name of a port defined on the Communications Server for Linux system.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc

PORT_ACTIVE
The specified port cannot be modified because it is currently active.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

delete_rcf_access

The **delete_rcf_access** command deletes access to the Communications Server for Linux Remote Command Facility (RCF) that was previously specified using **define_rcf_access**. For more information about RCF, refer to the *Communications Server for Linux Administration Guide*. This command prevents access to both SPCF and UCF. To allow access to one but prevent access to the other, use **define_rcf_access**.

Because RCF access parameters are defined as domain resources, this command is not associated with a particular node.

Communications Server for Linux acts on the RCF access parameters during node start-up; if RCF access is deleted while a node is running, the change does not take effect on the server where the node is running until the node is stopped and restarted.

Supplied Parameters

```
[delete_rcf_access]
```

No parameters are supplied for this command.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

delete_security_access_list

The **delete_security_access_list** command is used to do one of the following:

- Delete a security access list.
- Delete one or more users from a security access list but leave the list configured.

You can delete a user name from the security access list regardless of whether there are active conversations that were set up using that user name. Deleting the user name does not affect the active conversations, but the invoking program will not be able to set up any further conversations using the deleted user name.

Supplied Parameters

```
Parameter name           Type           Length      Default
[delete_security_access_list]
list_name                character      14

{security_user_name}
user_name                character      10
```

Supplied parameters are:

list_name
The name of the security access list being deleted, or the list from which user names are being deleted. This name is a string of 1–14 locally displayable characters which must match a previously defined security access list name.

To delete the complete security access list, do not specify any user names. To delete one or more user names from the list but leave the list configured, specify a `security_user_name` subrecord for each user name to be deleted, with the following information:

user_name
The user name being deleted. This must match a user name that is currently defined for this security access list.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
`PARAMETER_CHECK`

secondary_rc
Possible values are:

`INVALID_LIST_NAME`
The specified security access list name was not defined as a security access list name.

`INVALID_USER_NAME`
One or more of the specified user names did not match the name of a user defined for this security access list.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

delete_tn3270_access

The **delete_tn3270_access** command is used to do one of the following:

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

Supplied Parameters

Parameter name	Type	Length	Default
[delete_tn3270_access]			
default_record	constant		NO
client_address	character	256	(null string)
delete_options	constant		DELETE_USER
{tn3270_session_name}			
port_number	decimal		
listen_local_address	character	45	(null string)

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

Supplied parameters are:

default_record

Specifies whether **delete_tn3270_access** deletes the default access record. The default access record is used by a client whose TCP/IP address does not match any specific TN3270 access record. Deleting this record means that these clients cannot access TN server. Possible values are:

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

NO This command refers to a specific TN3270 access record specified in the *client_address* parameter.

client_address

The TCP/IP address of the client to be deleted, as specified on the **define_tn3270_access** command. This can be any of the following.

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

delete_options

To delete one or more sessions but not all the sessions, do not specify this parameter. Specify the sessions to be deleted using *port_number* parameters. To delete all sessions, specify one of the following values:

ALL_SESSIONS
Delete all sessions but leave the TN3270 client configured.

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

Each `tn3270_session_name` subrecord contains the following parameters:

port_number
The TCP/IP port number used for the session. If the *delete_options* parameter is not specified, use this parameter to specify each session to be deleted.

listen_local_address
The address on the local TN Server computer to which TN3270 clients connect. This parameter is optional.

- If this parameter was not specified when configuring the session, do not specify it on this command.
- If the address was specified when configuring the session, specify the same address on this command.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
`PARAMETER_CHECK`

secondary_rc

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

INVALID_PORT_NUMBER
The TCP/IP port number specified in the *port_number* parameter did not match any TCP/IP port number defined for this user.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

delete_tn3270_association

The **delete_tn3270_association** command deletes an association between a display LU and a printer LU, given the display LU name.

Supplied Parameters

```
Parameter                Type          Length
[delete_tn3270_association]
display_lu_name          character     8
```

Supplied parameter is:

display_lu_name
Name of the display LU whose association is to be deleted. This name is a 1–8 character string.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

INVALID_LU_NAME
The value specified for the *display_lu_name* parameter was not a valid type-A string.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc

INVALID_LU_NAME
No association is defined for the specified display LU.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

delete_tn_redirect

The **delete_tn_redirect** command is used to delete a Telnet client using the TN Redirector function, so that this user can no longer use TN Redirector to access a host.

Supplied Parameters

Parameter name	Type	Length	Default
[delete_tn_redirect]			
default_record	constant		NO
client_address	character	256	(null string)
client_port	decimal		
listen_local_address	character	45	(null string)

Supplied parameters are:

default_record
Specifies whether **delete_tn_redirect** deletes the default access record. The default access record is used by a client whose TCP/IP address does not match any specific TN Redirector access record. Possible values are:

YES This command refers to the default TN Redirector access record. The *client_address* parameter is not used.

NO This command refers to a specific TN Redirector access record specified in the *client_address* parameter.

client_address
The TCP/IP address of the client to be deleted. This can be any of the following.

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

If you use a name or alias, the Linux system 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).

client_port
The TCP/IP port number used by the client.

listen_local_address
The address on the local TN Server computer to which TN3270 clients connect. This parameter is optional.

- If this parameter was not specified when configuring the redirection record, do not specify it on this command.
- If the address was specified when configuring the redirection record, specify the same address on this command.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

INVALID_CLIENT_ADDRESS
The specified addressing information did not match any defined TN Redirector user.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

delete_tp

The **delete_tp** command deletes a TP definition.

Supplied Parameters

Parameter name	Type	Length
[delete_tp]		
tp_name	character	64

Supplied parameter is:

tp_name
Name of the TP to be deleted.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_TP_NAME
The *tp_name* parameter did not match the name of a defined TP.

SYSTEM_TP_CANT_BE_DELETED
The specified TP name is a TP name used internally by Communications Server for Linux; you cannot delete it.

State Check
No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions
Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

delete_tp_load_info

The **delete_tp_load_info** command is used to delete a TP load information entry. Both the *tp_name* and *lualias* parameters are mandatory. To delete all the entries for a particular TP, an application must first call the **query_tp_load_info**command for that TP and then delete the entries for the different LU aliases one at a time..

Supplied Parameters

Parameter name [delete_tp_load_info]	Type	Length	Default
tp_name	character	64	
lualias	character	8	(null string)

Supplied parameters are:

tp_name
The TP name of the TP load info entry to be deleted. This name is a 64-byte string.

lualias The LU alias of the TP load info entry to be deleted. This alias is an 8-byte string.

This parameter can be used only if the TP is an APPC application; it must not be specified if the TP is a CPI-C application.

Returned Parameters
No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes
If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check
If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_TP_NAME
The name specified for the *tp _name* parameter did not match the TP name of any defined TP load info entry.

INVALID_LU_ALIAS
The alias specified for the *lualias* parameter did not match any LU alias defined for a TP load info entry for the specified TP name.

State Check
No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions
Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

delete_userid_password

The **delete_userid_password** command deletes a password associated with a user ID, or deletes profiles for a user ID and password.

Supplied Parameters

Parameter name [delete_userid_password]	Type	Length	Default
delete_type	constant		REMOVE_USER
user_id	character	10	
profile	character	10	(null string)

(When you are deleting profiles without deleting the user, you can specify up to ten *profile* parameters.)

Supplied parameters are:

delete_type
Specifies the type of information to be deleted. Possible values are:

REMOVE_USER
Delete the user, password, and all associated profiles.

REMOVE_PROFILES
Delete only the specified profiles.

user_id User identifier. This ID is a type-AE character string.

profile Profiles associated with the user. Each profile is a type-AE character string.

Returned Parameters
No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

NO_PROFILES
The *delete_type* parameter was set to REMOVE_PROFILES, but no profiles were specified.

UNKNOWN_USER
The *user_id* parameter did not match a defined user ID.

INVALID_UPDATE_TYPE
The *delete_type* parameter was set to a value that was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

init_node

The **init_node** command starts the local node. It must be issued to a server where the node is not running. The Communications Server for Linux software must be started on the computer containing the node.

Supplied Parameters

[init_node]

No parameters are supplied for this command.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_NODE_NAME
The node name specified in the configuration file does not match the name of the Communications Server for Linux computer to which the command was issued.

NOT_SERVER
The node name specified in the configuration file matches the name of the Communications Server for Linux computer, but the specified computer is a client (not a server) and cannot run the node.

DLUR_NOT_SUPPORTED
The configuration of the node specifies that DLUR is supported, but the node is defined as a LEN node. DLUR cannot be supported on a LEN node.

State Check
If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

NODE_ALREADY_STARTED
The node name specified in the configuration file has already been started.

RESOURCE_NOT_LOADED
The node was not started because Communications Server for Linux detected one or more errors while attempting to load its configuration. Check the error log file for details about the errors.

INVALID_VERSION
The node was not started because there was a version mismatch between components of the Communications Server for Linux software. If you have upgraded your Communications Server for Linux license to include additional functions or users, check that you are using the correct version of the licensing software.

Other Conditions
Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

initialize_session_limit

The **initialize_session_limit** command initializes the session limits for a combination of local LU, partner LU, and mode. This command must be issued to a running node.

You must issue this command before you issue an **activate_session** command.

This command can be issued from a client. If it is issued from an AIX or Linux client, the command must run with the userid `root`, or with a userid that is a member of the `sys` group (AIX) or `sna` group (Linux).

Supplied Parameters

Parameter name [initialize_session_limit]	Type	Length	Default
lu_name	character	8	(null string)
lu_alias	character	8	(null string)
plu_alias	character	8	(null string)
fqplu_name	character	17	(null string)
mode_name	character	8	
set_negotiable	constant		NO
plu_mode_session_limit	decimal		
min_conwinners_source	decimal		0
min_conwinners_target	decimal		0
auto_act	decimal		0

Supplied parameters are:

lu_name
: LU name of the local LU. This name is a type-A character string. To indicate that the LU is defined by its LU alias instead of its LU name, do not specify this parameter.

lu_alias
: LU alias of the local LU. This alias is a character string using any locally displayable characters. This parameter is used only if *lu_name* is not specified.

  If *lu_name* and *lu_alias* are not specified, the command is forwarded to the LU associated with the CP (the default LU).

plu_alias
: LU alias of the partner LU. This alias is a character string using any locally displayable characters. To indicate that the partner LU is defined by its fully qualified LU name instead of its LU alias, do not specify this parameter.

fqplu_name
: Fully qualified name of the partner LU. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character partner LU name.

  This parameter is used only if the *plu_alias* parameter is not specified; it is ignored if *plu_alias* is specified.

mode_name
: Name of the mode to be used by the LUs. This name is a type-A character string starting with a letter.

set_negotiable
: Specifies whether the maximum negotiable session limit for this mode, as defined by **define_mode**, should be modified. Possible values are:

  YES Use the value specified by *plu_mode_session_limit* as the maximum negotiable session limit for this LU-LU-mode combination.

  NO Leave the maximum negotiable session limit as the value specified for the mode.

plu_mode_session_limit
: Requested total session limit for this LU-LU-mode combination; the

maximum number of parallel sessions allowed between these two LUs using this mode. This value will be negotiated with the partner LU. Specify a value in the range 1–32,767 (which must not exceed the session limit specified for the local LU on the **define_local_lu** command).

min_conwinners_source
Minimum number of sessions using this mode for which the local LU is the contention winner. The sum of the *min_conwinners_source* and *min_conwinners_target* parameters must not exceed the *plu_mode_session_limit* parameter. Specify a value in the range 0–32,767.

min_conwinners_target
Minimum number of sessions using this mode for which the partner LU is the contention winner. The sum of the *min_conwinners_source* and *min_conwinners_target* parameters must not exceed the *plu_mode_session_limit* parameter. Specify a value in the range 0–32,767.

auto_act
Number of contention winner sessions to be automatically activated after the session limits for the LU-LU-mode combination have been negotiated. If the negotiation of limits results in a number of contention winner sessions that is less than the value specified in this parameter, the actual number of sessions activated is less than the *auto_act* parameter value. Specify a value in the range 0–32,767 (which must not exceed the *plu_mode_session_limit* parameter or the session limit specified for the local LU on the **define_local_lu** command).

Returned Parameters

If the command executes successfully, the following parameters are returned:

primary_rc
OK

secondary_rc
Possible values are:

AS_NEGOTIATED
The session limits were initialized, but one or more values were negotiated by the partner LU.

AS_SPECIFIED
The session limits were initialized as requested, without being negotiated by the partner LU.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

EXCEEDS_MAX_ALLOWED
The *plu_mode_session_limit*, *min_conwinners_source*, *min_conwinners_target*, or *auto_act* parameter was set to a value outside the valid range.

CANT_CHANGE_TO_ZERO
The *plu_mode_session_limit* parameter cannot be set to `0` (zero) using this command; use the **reset_session_limit** command instead.

INVALID_LU_ALIAS
The *lu_alias* parameter did not match any defined local LU alias.

INVALID_LU_NAME
The *lu_name* parameter did not match any defined local LU name.

INVALID_MODE_NAME
The *mode_name* parameter did not match any defined mode name.

INVALID_PLU_NAME
The *fqplu_name* parameter did not match any defined partner LU name.

INVALID_SET_NEGOTIABLE
The *set_negotiable* parameter was not set to a valid value.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
`STATE_CHECK`

secondary_rc

MODE_NOT_RESET
One or more sessions are currently active for this LU-LU-mode combination. Use **change_session_limit** instead of **initialize_session_limit** to specify the limits.

Other Conditions

primary_rc

ALLOCATION_ERROR
Session limits could not be initialized because Communications Server for Linux was unable to allocate a session to the partner LU in order to negotiate the limits. Check the error log files for messages indicating the cause of this failure and take any action required.

secondary_rc

ALLOCATION_FAILURE_NO_RETRY
Session limits could not be initialized because Communications Server for Linux was unable to allocate a session to the partner LU in order to negotiate the limits. Check the error log files for messages indicating the cause of this failure and take any action required. Do not attempt to retry the command until the condition has been corrected.

primary_rc

CONV_FAILURE_NO_RETRY
The session limits could not be initialized because of a condition

that requires action (such as a configuration mismatch or a session protocol error). Check the Communications Server for Linux log file for information about the error condition, and correct it before retrying this command.

primary_rc

CNOS_PARTNER_LU_REJECT
Session limits could not be initialized because the node failed to successfully negotiate the limits with the partner LU. Check configuration at both the local LU and the partner LU.

secondary_rc

CNOS_COMMAND_RACE_REJECT
The command failed because the specified mode was being accessed by another administration program (or internally by the Communications Server for Linux software) for session activation or deactivation, or for session limit processing. Retry the command.

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

path_switch

The **path_switch** command requests that Communications Server for Linux switch a currently active Rapid Transport Protocol (RTP) connection to another path. If Communications Server for Linux cannot find a better path, it leaves the connection unchanged.

Supplied Parameters

```
Parameter name         Type        Length
[path_switch]
rtp_connection_name    character   8
```

Supplied parameter is:

rtp_connection_name
The RTP connection for which a change in path is requested.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

INVALID_RTP_CONNECTION_NAME_SPECIFIED
The value specified for the *rtp_connection_name* parameter did not match the name of an existing RTP connection.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc

PATH_SWITCH_IN_PROGRESS
Communications Server for Linux is currently changing the path for the RTP connection specified by the *rtp_connection_name* parameter.

Path Switch Disabled

If the command does not execute because the RTP partner node has disabled path switch by setting the path switch timer to zero, Communications Server for Linux returns the following parameter:

primary_rc
PATH_SWITCH_DISABLED

secondary_rc
(No secondary return code is returned.)

Path Switch Failure

If the command does not execute because the path switch attempt fails, Communications Server for Linux returns the following parameter:

primary_rc
UNSUCCESSFUL

secondary_rc
(No secondary return code is returned.)

Function Not Supported

If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameters:

primary_rc

FUNCTION_NOT_SUPPORTED
This node is not defined to support High Performance Routing (HPR).

secondary_rc
(This parameter is not used.)

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_active_transaction

The **query_active_transaction** command returns information about active multiple-domain support (MDS) transactions known to the Communications Server for Linux Management Services component. An active transaction is an MDS request for which a reply has not yet been received.

This command may be used to obtain information about a single transaction or about multiple transactions, depending on the options used. It must be issued to a running node.

Supplied Parameters

Parameter name [query_active_transaction]	Type	Length	Default
num_entries	decimal		1
list_options	constant		LIST_INCLUSIVE
fq_req_loc_cp_name	character	17	(null string)
req_agent_appl_name	character	8	(null string)
seq_num_dt	hex array	17	(null array)

Supplied parameters are:

num_entries
Maximum number of transactions for which data should be returned. You can specify 1 to return data for a specific transaction, a number greater than 1 to return data for multiple transactions, or 0 (zero) to return data for all transactions.

list_options
The position in the list of transactions from which Communications Server for Linux begins to return data. The list is ordered by *fq_req_loc_cp_name*, then by *req_agent_appl_name*, and finally in numerical order of *seq_num_dt*.

Possible values are:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the *fq_req_loc_cp_name*, *req_agent_appl_name*, and *seq_num_dt* parameters

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *fq_req_loc_cp_name*, *req_agent_appl_name*, and *seq_num_dt* parameters

fq_req_loc_cp_name
Fully qualified control point name of the transaction requester. This parameter is ignored if *list_options* is set to FIRST_IN_LIST. The name is a type-A character string. It consists of a 1–8 character network name, followed by a period, followed by a 1–8 character control point name.

req_agent_appl_name
Application name of the transaction requestor. This parameter is ignored if *list_options* is set to FIRST_IN_LIST.

This name is normally a type-1134 character string (using uppercase A–Z and numerals 0–9); alternatively, it can be one of the MS Discipline-Specific Application Programs specified in *Systems Network Architecture: Management Services*.

seq_num_dt
Sequence number date/time correlator (17 bytes long) of the original transaction, as defined in *Systems Network Architecture: Formats*. This parameter is ignored if *list_options* is set to FIRST_IN_LIST.

Returned Parameters

Parameter name	Type	Length
fq_origin_cp_name	character	17
origin_ms_appl_name	character	8
fq_dest_cp_name	character	17
dest_ms_appl_name	character	8
fq_req_loc_cp_name	character	17
req_agent_appl_name	character	8
seq_num_dt	hex array	17

If the command executes successfully, Communications Server for Linux returns the following parameters:

fq_origin_cp_name
Fully qualified control point name of the CP initiating the transaction.

origin_ms_appl_name
Name of the application from which the transaction originates. This name is usually a type-1134 character string; alternatively, it can also be one of the MS Discipline-Specific Application Programs specified in *Systems Network Architecture: Management Services*.

fq_dest_cp_name
Fully qualified control point name of the transaction destination.

dest_ms_appl_name
Application name of the destination application for the transaction. This name is usually a type-1134 character string; alternatively, it can be one of the MS Discipline-Specific Application Programs specified in *Systems Network Architecture: Management Services*.

fq_req_loc_cp_name
Fully qualified control point name of the transaction requestor.

req_agent_appl_name
Application name of the transaction requester. This name is usually a type-1134 character string; alternatively, it can be one of the MS Discipline-Specific Application Programs specified in *Systems Network Architecture: Management Services*.

seq_num_dt
Sequence number date/time correlator (17 bytes long) of the original transaction, as defined in *Systems Network Architecture: Formats*.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_ACTIVE_TRANSACTION
The *fq_req_loc_cp_name*, *req_agent_appl_name*, and *seq_num_dt* parameter values did not match the parameter values specified for an active transaction.

State Check
No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Function Not Supported
If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameters:

primary_rc

FUNCTION_NOT_SUPPORTED
The local node does not support MS network management functions; support is defined by the *mds_supported* parameter on the **define_node** command.

secondary_rc
(This parameter is not used.)

Other Conditions
Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

query_adjacent_nn

The **query_adjacent_nn** command returns information about adjacent network nodes (the network nodes to which CP-CP sessions are active or have been active at some time). The command can be used only if the Communications Server for Linux node is a network node (NN); it is not valid if the node is an end node (EN) or low-entry networking (LEN) node.

This command can be used to obtain information about a specific adjacent network node or about multiple adjacent network nodes, depending on the options used. It must be issued to a running node.

Supplied Parameters

```
Parameter name     Type        Length    Default
[query_adjacent_nn]
num_entries        decimal     1
list_options       constant              LIST_INCLUSIVE
adj_nncp_name      character   17        (null string)
```

Supplied parameters are:

num_entries
Maximum number of adjacent NNs for which data should be returned. You can specify 1 to return data for a specific adjacent NN, a number greater than 1 to return data for multiple adjacent NNs, or 0 (zero) to return data for all adjacent NNs.

list_options
The position in the list of adjacent NNs from which Communications Server for Linux begins to return data.

Possible values are:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the *adj_nncp_name* parameter

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *adj_nncp_name* parameter

adj_nncp_name
Fully qualified name of the adjacent NN for which information is required, or the name to be used as an index into the list of adjacent NNs. This value is ignored if *list_options* is set to FIRST_IN_LIST. Specify 3–17 type-A characters consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character adjacent NN name.

Returned Parameters

Parameter name	Type	Length
adj_nncp_name	character	17
cp_cp_sess_status	constant	
out_of_seq_tdus	decimal	
last_frsn_sent	decimal	
last_frsn_rcvd	decimal	

If the command executes successfully, the following parameters are returned:

adj_nncp_name
Fully qualified name of the adjacent NN.

cp_cp_sess_status
Status of the CP-CP session to the adjacent NN. Possible values are:

ACTIVE The session is active.

CONWINNER_ACTIVE
The session (a contention winner session) is active.

CONLOSER_ACTIVE
The session (a contention loser session) is active.

INACTIVE
The session is inactive.

out_of_seq_tdus
Number of out-of-sequence TDUs received from this node.

last_frsn_sent
The last flow reduction sequence number (FRSN) sent to this node.

last_frsn_rcvd
The last flow reduction sequence number received from this node.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_ADJ_NNCP_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *adj_nncp_name* parameter value was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Function Not Supported

If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameters:

primary_rc

FUNCTION_NOT_SUPPORTED
The local node is an end node or LEN node. This command is valid only for a network node.

secondary_rc
(This parameter is not used.)

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_available_tp

The **query_available_tp** command returns information about active invokable transaction programs (TPs). Active invokable TPs are APPC applications that have issued the RECEIVE_ALLOCATE verb, or CPI-C applications that have issued the Accept_Conversation or Accept_Incoming call. This command can be used to obtain information about a specific TP or about multiple TPs, depending on the options used. It returns information about all active invokable TPs that are running, whether or not they currently have an APPC verb or a CPI-C call outstanding to accept an incoming conversation.

This command must be issued to a running node.

Supplied Parameters

Parameter name [query_available_tp]	Type	Length	Default
num_entries	decimal		1
list_options	constant		LIST_INCLUSIVE
tp_name	character	64	(null string)
system_name	character	128	(null string)

Supplied parameters are:

num_entries
Maximum number of TPs for which data should be returned. You can specify 1 to return data for a specific TP, a number greater than 1 to return data for multiple TPs, or 0 (zero) to return data for all TPs.

list_options
The position in the list of TPs from which Communications Server for Linux begins to return data.

Possible values are:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the combination of the *tp_name* and *system_name* parameters

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the combination of the *tp_name* and *system_name* parameters

tp_name
TP name for which information is required. This name is a 64-byte string. This value is ignored if *list_options* is set to FIRST_IN_LIST.

system_name
The computer name for which TP information is required. The system name is a string of 1–64 locally displayable characters. This value is ignored if *list_options* is set to FIRST_IN_LIST.

If the computer name includes a . (period) character, Communications Server for Linux assumes that it is a fully-qualified name; otherwise it performs a DNS lookup to determine the computer name.

It is not necessary to specify the *system_name* parameter if Communications Server for Linux is standalone. In a client/server system, specify the system name to list only TPs on a specified computer. If you do not specify this parameter, Communications Server for Linux lists TPs on all computers.

Returned Parameters

Parameter name	Type	Length
tp_name	character	64
system_name	character	128

If the command executes successfully, Communications Server for Linux returns the following parameters:

tp_name
TP name.

system_name
Name of the computer where the TP is running.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

UNKNOWN_TP
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *tp_name* parameter value was not valid, or the *system_name* parameter was supplied and was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

query_buffer_availability

The **query_buffer_availability** command returns information about the amount of STREAMS buffer space that Communications Server for Linux is currently using, the maximum amount it has used, and the maximum amount available (specified using the **set_buffer_availability** command). This information enables you to check STREAMS buffer usage and set the limit appropriately to ensure that sufficient buffer space is available for Communications Server for Linux components and for other programs on the Linux computer. The command also returns additional internal values related to buffer usage for use by Communications Server for Linux support personnel.

Supplied Parameters

Parameter name	Type	Length	Default
[query_buffer_availability]			
reset_max_values	constant		NO

Supplied parameter is:

reset_max_values
Specify whether Communications Server for Linux should reset the *max_** values (after returning them on this command) to match the current values of these parameters. This ensures that a subsequent **query_buffer_availability** command will return the maximum values reached since this command was issued, rather than the maximum values reached since the system was started (or since the *max_** values were last reset). Possible values are:

YES Reset the *max_** values to match the current values.

NO Do not reset the *max_** values.

Returned Parameters

Parameter name	Type	Length
buf_avail	decimal	
buf_total_count	decimal	
buf_total_bytes	decimal	
buf_rsrv_count	decimal	
buf_rsrv_bytes	decimal	
buf_res_use_count	decimal	
buf_res_use_bytes	decimal	
peak_usage	decimal	
peak_decay	decimal	
throttle_status	decimal	
buf_use_status	constant	
max_buf_total_count	decimal	
max_buf_total_bytes	decimal	
max_buf_rsrv_count	decimal	
max_buf_rsrv_bytes	decimal	
max_buf_res_use_count	decimal	
max_buf_res_use_bytes	decimal	
max_peak_usage	decimal	
max_throttle_status	decimal	
max_buf_use_status	constant	
debug_param	character	32

If the command executes successfully, Communications Server for Linux returns the following parameters:

buf_avail
: The maximum amount of STREAMS buffer space available to Communications Server for Linux, in bytes, as defined by a **set_buffer_availability** command.

buf_total_count
: The total number of buffers currently allocated to Communications Server for Linux components.

buf_total_bytes
: The total amount of buffer storage, in bytes, currently allocated to Communications Server for Linux components.

buf_rsrv_count
: The total number of buffers reserved.

buf_rsrv_bytes
: The total amount of storage in buffers reserved, in bytes.

buf_res_use_count
: The number of reserved buffers in use.

buf_res_use_bytes
: The number of bytes in the reserved buffers currently in use.

peak_usage
: Peak buffer usage—smoothed percentage of buffers that are actually used.

peak_decay
: Smoothing parameter.

throttle_status
: Adaptive pacing status.

buf_use_status
: Congestion status. Possible values are:

```
CONGESTED
UNCONGESTED
```

max_buf_total_count
The maximum number of buffers that have been allocated to Communications Server for Linux components at any one time.

max_buf_total_bytes
The maximum amount of buffer storage that has been allocated to Communications Server for Linux components at any one time.

max_buf_rsrv_count
The maximum number of buffers that can be reserved.

max_buf_rsrv_bytes
The maximum amount of buffer storage that can be reserved, in bytes.

max_buf_res_use_count
The maximum number of reserved buffers that can be in use.

max_buf_res_use_bytes
The maximum number of bytes of reserved buffers that can be in use at any time.

max_peak_usage
Maximum peak buffer usage—smoothed percentage of buffers actually used.

max_throttle_status
Maximum adaptive pacing status.

max_buf_use_status
Maximum congestion status. Possible values are:

```
CONGESTED
UNCONGESTED
```

debug_param
This parameter is provided for use by Communications Server for Linux support personnel.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_central_logger

The **query_central_logger** command returns the name of the node currently defined as the central logger. The central logger is the node holding the central log file to which Communications Server for Linux log messages from all servers are sent. This command does not return information about whether central logging is active; use **query_central_logging** to determine this.

This command must be issued without specifying a node name.

Supplied Parameters

```
[query_central_logger]
```

No parameters are supplied for this command.

Returned Parameters

Parameter name	Type	Length
node_name	character	128

If the command executes successfully, Communications Server for Linux returns the following parameter:

node_name
The name of the node defined as the central logger. You can issue **query_central_logging** to this node to determine whether central logging is currently enabled.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc

NO_CENTRAL_LOG
No master server is currently active.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_central_logging

The **query_central_logging** command returns information about whether Communications Server for Linux log messages are sent to a central file from all servers, or to a separate file on each server. For more information about log files, see "set_log_file" on page 553.

This command must be issued without specifying a node name.

Supplied Parameters

```
[query_central_logging]
```

No parameters are supplied for this command.

Returned Parameters

```
Parameter name          Type
enabled                 constant
```

If the command executes successfully, Communications Server for Linux returns the following parameter:

enabled Specifies whether central logging is enabled or disabled. Possible values are:

YES Central logging is enabled. All log messages are sent to a single central file on the node that is currently the central logger.

NO Central logging is disabled. Log messages from each server are sent to a file on that server (specified using the **set_log_file** command).

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc

PARAMETER_CHECK

secondary_rc

NOT_CENTRAL_LOGGER
The command was issued to a specific node. It must be issued without specifying a node name.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_cn

The **query_cn** command returns information about adjacent connection networks. This command can be used to obtain information about a specific connection network or about multiple connection networks, depending on the options used.

The command is valid only at a network node or an end node; it is not valid at a LEN node.

Supplied Parameters

Parameter name	Type	Length	Default
[query_cn]			
num_entries	decimal		1
list_options	constant		LIST_INCLUSIVE
fqcn_name	character	17	(null string)

Supplied parameters are:

num_entries
: Maximum number of CNs for which data should be returned. You can specify 1 to return data for a specific CN, a number greater than 1 to return data for multiple CNs, or 0 (zero) to return data for all CNs.

list_options
: The position in the list of CNs from which Communications Server for Linux begins to return data.

  Possible values are:

  FIRST_IN_LIST
  : Start at the first entry in the list

  LIST_INCLUSIVE
  : Start at the entry specified by the *fqcn_name* parameter

  LIST_FROM_NEXT
  : Start at the entry immediately following the entry specified by the *fqcn_name* parameter

fqcn_name
: Fully qualified name of the CN for which information is required, or the name to be used as an index into the list of CNs. This value is ignored if *list_options* is set to FIRST_IN_LIST. Specify 3–17 type-A characters consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character CN name.

Returned Parameters

Parameter name	Type	Length
fqcn_name	character	17
num_act_ports	decimal	
description	character	31
num_ports	decimal	
effect_cap	decimal	
connect_cost	decimal	
byte_cost	decimal	
security	constant	
prop_delay	constant	
user_def_parm_1	decimal	
user_def_parm_2	decimal	
user_def_parm_3	decimal	

If the command executes successfully, the following parameters are returned:

fqcn_name
Fully qualified name of the CN.

num_act_ports
The number of active ports on the connection network.

description
A text string describing the CN, as specified in the definition of the CN.

num_ports
The total number of ports on the connection network.

effect_cap
A decimal value representing the line speed in bits per second.

connect_cost
Cost per connect time.

byte_cost
Cost per byte.

security
Security level of the network. Possible values are:

SEC_NONSECURE
No security.

SEC_PUBLIC_SWITCHED_NETWORK
Data is transmitted over a public switched network.

SEC_UNDERGROUND_CABLE
Data is transmitted over secure underground cable.

SEC_SECURE_CONDUIT
Data is transmitted over a line in a secure conduit that is not guarded.

SEC_GUARDED_CONDUIT
Data is transmitted over a line in a conduit that is protected against physical tapping.

SEC_ENCRYPTED
Data is encrypted before transmission over the line.

SEC_GUARDED_RADIATION
Data is transmitted over a line that is protected against physical and radiation tapping.

prop_delay
Propagation delay (the time that a signal takes to travel the length of the link). Specify one of the following values, according to the type of link:

PROP_DELAY_MINIMUM
Minimum propagation delay.

PROP_DELAY_LAN
Delay is less than .5 microseconds (typical for a LAN).

PROP_DELAY_TELEPHONE
Delay is in the range .5–50 microseconds (typical for a telephone network).

PROP_DELAY_PKT_SWITCHED_NET
Delay is in the range 50–250 microseconds (typical for a packet-switched network).

PROP_DELAY_SATELLITE
Delay is greater than 250 microseconds (typical for a satellite link).

PROP_DELAY_MAXIMUM
Maximum propagation delay.

user_def_parm_1 **through** *user_def_parm_3*
User-defined parameters that you can use to include other characteristics not covered by the above parameters. Each of these parameters has a value in the range 0–255.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_CN_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the value specified in the *fqcn_name* parameter was not valid.

State Check

No specific state errors, other than those listed in Appendix A, “Common Return Codes from snaadmin Commands,” on page 589, occur for this command.

Function Not Supported

If the command does not execute because the node’s configuration does not support it, Communications Server for Linux returns the following parameters:

primary_rc

FUNCTION_NOT_SUPPORTED
The local node is a LEN node. This command is valid only at a network node or an end node.

secondary_rc
(This parameter is not used.)

Other Conditions

Appendix A, “Common Return Codes from snaadmin Commands,” on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_cn_port

The **query_cn_port** command returns information about ports defined on adjacent connection networks. This command can be used to obtain information about a specific port or about multiple ports, depending on the options used.

The command is valid only at a network node or an end node; it is not valid at a LEN node.

Supplied Parameters

Parameter name	Type	Length	Default
[query_cn_port]			
num_entries	decimal		1
list_options	constant		LIST_INCLUSIVE
fqcn_name	character	17	
port_name	character	8	(null string)

Supplied parameters are:

num_entries
: Maximum number of ports for which data should be returned. You can specify 1 to return data for a specific port, a number greater than 1 to return data for multiple ports, or 0 (zero) to return data for all ports.

list_options
: The position in the list of ports from which Communications Server for Linux begins to return data.

  Possible values are:

  FIRST_IN_LIST
  : Start at the first entry in the list

  LIST_INCLUSIVE
  : Start at the entry specified by the *port_name* parameter

  LIST_FROM_NEXT
  : Start at the entry immediately following the entry specified by the *port_name* parameter

fqcn_name
: Fully qualified name of the CN on which the required port is defined, or the name to be used as an index into the list of CNs and ports. Specify 3–17 type-A characters consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character CN name. This parameter must always be set.

port_name
: Name of the port for which information is required, or the name to be used as an index into the list of ports. This name is a string of 1–8 characters. This parameter is ignored if *list_options* is set to FIRST_IN_LIST.

Returned Parameters

Parameter name	Type	Length
fqcn_name	character	17
port_name	character	8
tg_num	decimal	

If the command executes successfully, the following parameters are returned:

fqcn_name
Fully qualified name of the CN.

port_name
Name of the port.

tg_num
Transmission group number for the specified port.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_CN_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the value specified in the *fqcn_name* parameter was not valid.

INVALID_PORT_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the value specified in the *port_name* parameter was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Function Not Supported

If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameters:

primary_rc

FUNCTION_NOT_SUPPORTED
The local node is a LEN node. This command is valid only at a network node or an end node.

secondary_rc
(This parameter is not used.)

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_conversation

The **query_conversation** command returns information about conversations using a particular local LU. This command can be used to obtain information about a specific conversation or a range of conversations, depending on the options used.

This command must be issued to a running node.

Supplied Parameters

Parameter name	Type	Length	Default
[query_conversation]			
num_entries	decimal		1
list_options	constant		SUMMARY + LIST_INCLUSIVE
conv_id	hex number	4	0x00
lu_name	character	8	(null string)
lu_alias	character	8	(null string)
session_id	hex array	8	(null array)

Supplied parameters are:

num_entries
: Maximum number of sessions for which data should be returned. You can specify 1 to return data for a specific conversation, a number greater than 1 to return data for multiple conversations, or 0 to return data for all conversations.

list_options
: The position in the list of conversations from which Communications Server for Linux begins to return data. Specify one of the following values:

  FIRST_IN_LIST
  : Start at the first entry in the list

  LIST_INCLUSIVE
  : Start at the entry specified by the combination of local LU, partner LU, and conversation ID

  LIST_FROM_NEXT
  : Start at the entry immediately following the entry specified by the combination of local LU, partner LU, and conversation ID

conv_id
: The identifier of the conversation for which information is required, or the conversation ID to be used as an index into the list of conversations. This parameter is ignored if *list_options* is set to FIRST_IN_LIST.

lu_name
: LU name of the local LU. This name is a type-A character string. To indicate that the LU is identified by its LU alias instead of its LU name, do not specify this parameter. To specify the LU associated with the local CP (the default LU), do not specify either *lu_name* or *lu_alias*.

lu_alias
: Locally defined LU alias. This parameter is used only if *lu_name* is not specified. To specify the LU associated with the local CP (the default LU), do not specify either *lu_name* or *lu_alias*.

session_id
: 8-byte session identifier. To list only information about conversations associated with a specific session, specify the session identifier. To obtain a complete list for all sessions, do not specify this parameter.

Returned Parameters

Parameter name	Type	Length
conv_id	hex number	4
local_tp_name	character	64
partner_tp_name	character	64
tp_id	hex array	8
sess_id	hex array	8
conv_start_time	decimal	
bytes_sent	decimal	
bytes_received	decimal	
conv_state	constant	
duplex_type	constant	

If the command executes successfully, Communications Server for Linux returns the following parameters:

conv_id
: Conversation identifier. The conversation ID was returned by the ALLOCATE verb in the invoking TP, or by the RECEIVE_ALLOCATE verb in the invoked TP.

local_tp_name
: The name of the local TP in the conversation.

partner_tp_name
: The name of the partner TP in the conversation. This parameter is returned only if the conversation was started by the local TP; it is reserved if the conversation was started by the remote TP.

tp_id
: The TP identifier of the conversation.

sess_id
: The session identifier of the session allocated to the conversation.

conv_start_time
: The elapsed time in hundredths of seconds between the time when the Communications Server for Linux node was started and the time when the conversation was started.

bytes_sent
: The number of bytes that have been sent from the local TP to the partner TP since the start of the conversation.

bytes_received
: The number of bytes that have been received from the partner TP by the local TP since the start of the conversation.

conv_state
: The current state of the conversation. Values for a half-duplex conversation:
  - CONFIRM
  - CONFIRM_DEALL
  - CONFIRM_SEND
  - END_CONV
  - PEND_DEALL
  - PEND_POST
  - POST_ON_RECEIPT
  - RECEIVE
  - RESET
  - SEND

- SEND_PENDING

Values for a full-duplex conversation:

- RESET
- SEND_ONLY
- SEND_RECEIVE
- RECEIVE_ONLY

duplex_type
The duplex type of the conversation. Values:

- HALF_DUPLEX
- FULL_DUPLEX

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

BAD_CONV_ID
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied conversation ID, but the *conv_id* parameter value was not valid.

INVALID_LU_ALIAS
The *lu_alias* parameter value was not valid.

INVALID_LU_NAME
The *lu_name* parameter value was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_cos

The **query_cos** command returns route calculation information for a specific class of service (COS). This command can be used to obtain information about a specific COS or about multiple COSs, depending on the options used.

Supplied Parameters

Parameter name	Type	Length	Default
[query_cos]			
num_entries	decimal		1
list_options	constant		LIST_INCLUSIVE
cos_name	character	8	(null string)

Supplied parameters are:

num_entries
Maximum number of classes of service (COSs) for which data should be returned. You can specify 1 to return data for a specific COS, a number greater than 1 to return data for multiple COSs, or 0 (zero) to return data for all COSs.

list_options
The position in the list of COSs from which Communications Server for Linux begins to return data.

Possible values are:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the *cos_name* parameter

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *cos_name* parameter

cos_name
COS name for which data is required, or the name to be used as an index into the list. This value is ignored if *list_options* is set to FIRST_IN_LIST. The name is a type-A character string starting with a letter.

Returned Parameters

Parameter name	Type	Length
cos_name	character	8
description	character	31
transmission_priority	constant	
num_of_node_rows	decimal	
num_of_tg_rows	decimal	
trees	decimal	
calcs	decimal	
rejs	decimal	

If the command executes successfully, Communications Server for Linux returns the following parameters:

cos_name
Specifies the class of service (COS) name which is an 8-byte character string.

description
A text string describing the COS, as specified in the definition of the COS.

transmission_priority
Specifies the transmission priority. Possible values are:

LOW The session using this COS is given low priority.

MEDIUM The session using this COS is given medium priority.

HIGH The session using this COS is given high priority.

NETWORK
The session using this COS is given the highest priority.

num_of_node_rows
Number of node rows defined for this COS.

num_of_tg_rows
Number of TG rows defined for this COS.

trees Number of route tree caches built for this COS since the last initialization.

calcs Number of session activation requests (and therefore route calculations) specifying this COS.

rejs Number of session activation requests that failed because there was no acceptable route through the network from this node to the named destination. A route is only acceptable if it is made up entirely of active TGs and nodes that can provide the specified class of service.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_COS_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *cos_name* parameter value was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_cos_node_row

The **query_cos_node_row** command returns node row information for a specified class of service (COS). This command can be used to obtain information about a specific COS node row or about multiple COS node rows, depending on the options used.

Supplied Parameters

Parameter name	Type	Length	Default
[query_cos_node_row]			
num_entries	decimal		1
list_options	constant		LIST_INCLUSIVE
cos_name	character	8	(null string)
node_row_index	decimal		0

Supplied parameters are:

num_entries
: Maximum number of COS node rows for which data should be returned. You can specify 1 to return data for a specific COS node row, a number greater than 1 to return data for multiple COS node rows, or 0 (zero) to return data for all COS node rows.

list_options
: The position in the list of COS node rows from which Communications Server for Linux begins to return data. The list is ordered by *cos_name*, and then by *node_row_index*, for each COS.

  Possible values are:

  FIRST_IN_LIST
  : Start at the first entry in the list

  LIST_INCLUSIVE
  : Start at the entry specified by the combination of the *cos_name* and *node_row_index* parameters

  LIST_FROM_NEXT
  : Start at the entry immediately following the entry specified by the combination of the *cos_name* and *node_row_index* parameters

cos_name
: Class of service name for which node row information is required, or the name to be used as an index into the list. This value is ignored if *list_options* is set to FIRST_IN_LIST. The name is a type-A character string starting with a letter.

node_row_index
: Node row number for which information is required, or the number to be used as an index into the list. This value is ignored if *list_options* is set to FIRST_IN_LIST. Use **query_cos** to determine the number of node rows associated with this COS.

Returned Parameters

Parameter name	Type	Length
cos_name	character	8
node_row_index	decimal	
min_rar	decimal	
min_status	constant	
max_rar	decimal	
max_status	constant	
weight	decimal	

If the command executes successfully, Communications Server for Linux returns the following parameters:

cos_name
: Specifies the class of service (COS) name.

node_row_index
Specifies the node row index.

min_rar **through** *weight*
For more information about these parameters, see "define_cos" on page 36.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_COS_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *cos_name* parameter value was not valid.

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

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

query_cos_tg_row

The **query_cos_tg_row** command returns TG row information for a specified class of service (COS). This command can be used to obtain information about a specific COS TG row or about multiple COS TG rows, depending on the options used.

The information is returned as a formatted list. To obtain information about a specific TG row or to obtain the list information in several rows, specify values for the *tg_row_index* and *cos_name* parameters. This returned list is ordered by the *cos_name* first and then by *tg_row_index*. The *cos_name* is ordered first by name length and then by ASCII lexicographical ordering for names of the same length. The *tg_row_index* is ordered numerically.

Supplied Parameters

Parameter name	Type	Length	Default
[query_cos_tg_row]			
num_entries	decimal		1
list_options	constant		LIST_INCLUSIVE
cos_name	character	8	(null string)
tg_row_index	decimal		0

Supplied parameters are:

num_entries
: Maximum number of COS TG rows for which data should be returned. You can specify 1 to return data for a specific COS TG row, a number greater than 1 to return data for multiple COS TG rows, or `0` (zero) to return data for all COS TG rows.

list_options
: The position in the list of COS TG rows from which Communications Server for Linux begins to return data. The list is ordered by *cos_name*, and then by *tg_row_index*, for each COS.

  Possible values are:

  FIRST_IN_LIST
  : Start at the first entry in the list

  LIST_INCLUSIVE
  : Start at the entry specified by the combination of the *cos_name* and *tg_row_index* parameters

  LIST_FROM_NEXT
  : Start at the entry immediately following the entry specified by the combination of the *cos_name* and *tg_row_index* parameters

cos_name
: Class of service (COS) name for which data is required, or the name to be used as an index into the list. The name is a type-A character string starting with a letter. This parameter is ignored if *list_options* is set to `FIRST_IN_LIST`.

tg_row_index
: TG row number for which data is required, or the number to be used as an index into the list. The first row has an index of `0` (zero). This parameter is ignored if *list_options* is set to `FIRST_IN_LIST`.

Returned Parameters

Parameter name	Type	Length
cos_name	character	8
tg_row_index	decimal	
min_effect_cap	hex number	
min_connect_cost	decimal	
min_byte_cost	decimal	
min_security	constant	
min_prop_delay	constant	
min_user_def_parm_1	decimal	
min_user_def_parm_2	decimal	
min_user_def_parm_3	decimal	
max_effect_cap	hex number	
max_connect_cost	decimal	
max_byte_cost	decimal	
max_security	constant	
max_prop_delay	constant	
max_user_def_parm_1	decimal	
max_user_def_parm_2	decimal	
max_user_def_parm_3	decimal	
weight	decimal	

If the command executes successfully, Communications Server for Linux returns the following parameters:

cos_name
Specifies the class of service (COS) name.

tg_row_index
Specifies the TG row index (the first row has an index of zero).

min_effect_cap
Minimum limit for line speed, in bits per second.

min_connect_cost
Minimum limit for cost per connect time.

min_byte_cost
Minimum limit for cost per byte.

min_security
Minimum level of security. Possible values are:

SEC_NONSECURE
Data is transmitted over an unsecured network.

SEC_PUBLIC_SWITCHED_NETWORK
Data is transmitted over a public-switched network.

SEC_UNDERGROUND_CABLE
Data is transmitted over secure underground cable.

SEC_SECURE_CONDUIT
Data is transmitted over a line in a secure conduit that is not guarded.

SEC_GUARDED_CONDUIT
Data is transmitted over a line in a conduit that is protected against physical tapping.

SEC_ENCRYPTED
Data is encrypted before transmission over the line.

SEC_GUARDED_RADIATION
Data is transmitted over a line that is protected against physical and radiation tapping.

min_prop_delay
Minimum limits for propagation delay, which is the time that a signal takes to travel the length of the link, in microseconds. Possible values are:

PROP_DELAY_LAN
Delay less than .5 microseconds (typical for a LAN) or minimum propagation delay. This value is returned if the **define_cos** specified either PROP_DELAY_MINIMUM or PROP_DELAY_LAN.

PROP_DELAY_TELEPHONE
Delay is in the range .5–50 microseconds (typical delay for a telephone network).

PROP_DELAY_PKT_SWITCHED_NET
Delay is in the range 50–250 microseconds (typical delay for a packet-switched network).

PROP_DELAY_SATELLITE
Delay is greater than 250 microseconds (typical delay for a satellite link).

PROP_DELAY_MAXIMUM
Maximum propagation delay.

min_user_def_parm_1 **through** *min_user_def_parm_3*
Minimum limits for user-defined parameters, which you can use to include TG characteristics not covered by previously defined parameters. Each of these parameters has a range of 0–255.

max_effect_cap
Maximum limit for line speed, in bits per second.

max_connect_cost
Maximum limit for cost per connect time.

max_byte_cost
Maximum limit for cost per byte.

max_security
Maximum level of security. Possible values are:

SEC_NONSECURE
Data is transmitted over an unsecured network.

SEC_PUBLIC_SWITCHED_NETWORK
Data is transmitted over a public-switched network.

SEC_UNDERGROUND_CABLE
Data is transmitted over secure underground cable.

SEC_SECURE_CONDUIT
Data is transmitted over a line in a secure conduit that is not guarded.

SEC_GUARDED_CONDUIT
Data is transmitted over a line in a conduit that is protected against physical tapping.

SEC_ENCRYPTED
Data is encrypted before transmission over the line.

SEC_GUARDED_RADIATION
Data is transmitted over a line that is protected against physical and radiation tapping.

SEC_MAXIMUM
Data is transmitted over a network that has maximum security.

max_prop_delay
Maximum limits for propagation delay, which is the time that a signal takes to travel the length of the link, in microseconds.

Possible values are:

PROP_DELAY_MINIMUM
Minimum propagation delay.

PROP_DELAY_LAN
Delay is less than .5 microseconds (typical for a LAN).

PROP_DELAY_TELEPHONE
Delay is in the range .5–50 microseconds (typical delay for a telephone network).

PROP_DELAY_PKT_SWITCHED_NET
Delay is in the range 50–250 microseconds (typical delay for a packet-switched network).

PROP_DELAY_SATELLITE
Delay is greater than 250 microseconds (typical delay for a satellite link), or maximum propagation delay. This value is returned if the **define_cos** specified either PROP_DELAY_SATELLITE or PROP_DELAY_MAXIMUM for *max_prop_delay*.

max_user_def_parm_1 **through** *max_user_def_parm_3*
Maximum limits for user-defined parameters, which you can use to include TG characteristics not covered by previously defined parameters. Each of these parameters has a range of 0–255.

weight Weight associated with this TG row.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_COS_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *cos_name* parameter value was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_cpic_side_info

The **query_cpic_side_info** command returns the side information entry for a given symbolic destination name or for multiple symbolic destination names, depending on the options used.

Because CPI-C side information entries are defined as domain resources, this command is not associated with a particular node.

Supplied Parameters

Parameter name	Type	Length	Default
[query_cpic_side_info]			
num_entries	decimal		1
list_options	constant		LIST_INCLUSIVE
sym_dest_name	character	8	(null string)

Supplied parameters are:

num_entries
Maximum number of symbolic destination names for which data should be returned. You can specify 1 to return data for a specific symbolic destination name, a number greater than 1 to return data for multiple symbolic destination names, or 0 (zero) to return data for all symbolic destination names.

list_options
The position in the list of symbolic destination names from which Communications Server for Linux begins to return data.

Possible values are:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the *sym_dest_name* parameter

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *sym_dest_name* parameter

sym_dest_name
Symbolic destination name for which data is required, or the name to be used as an index into the list. Valid characters are uppercase A–Z and numerals 0–9. This value is ignored if *list_options* is set to FIRST_IN_LIST.

Returned Parameters

Parameter name	Type	Length
sym_dest_name	character	8
description	character	31
partner_lu_name	character	17
tp_name_type	constant	
tp_name	character	64
mode_name	character	8
conversation_security_type	constant	
security_user_id	character	10
security_password	character	10
lu_alias	character	8

If the command executes successfully, Communications Server for Linux returns the following parameters:

sym_dest_name
Symbolic destination name for the returned side information entry.

description
A text string describing the side information entry, as specified in the definition of the side information entry.

partner_lu_name **through** *lu_alias*
For more information about these parameters, see "define_cpic_side_info" on page 41.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check
No parameter errors occur for this command.

State Check
If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc

INVALID_SYM_DEST_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *sym_dest_name* parameter value was not valid.

Other Conditions
Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_cs_trace

The **query_cs_trace** command returns information about the current tracing options for data sent between computers in the Communications Server for Linux domain. For more information about tracing options, refer to *Communications Server for Linux Diagnostics Guide*.

This command may be issued to a running node or to a Remote API Client on AIX or Linux. To issue the command to a client computer, use the **snaadmin** program on the client computer without specifying a node name.

On Windows clients, client/server tracing is controlled by options in the Windows Registry. For more information, refer to *Communications Server for Linux Diagnostics Guide*.

Supplied Parameters

Parameter name	Type	Length	Default
[query_cs_trace]			
dest_sys	character	128	(null string)

Supplied parameter is:

dest_sys
The server name for which tracing options are to be queried.

To query tracing options on messages flowing between the computer to which this command is issued (either the local computer or one identified by the **-n** option on the **snaadmin** program) and one other server in the domain, specify the name of the other server.

If the server name includes a **.** (period) character, Communications Server for Linux assumes that it is a fully-qualified name; otherwise it performs a DNS lookup to determine the server name.

To query the default tracing options (set by a **set_cs_trace** command with no system name), do not specify this parameter.

Returned Parameters

Parameter name	Type
trace_flags	constant
trace_direction	constant

If the command executes successfully, Communications Server for Linux returns the following parameters:

trace_flags
The types of tracing currently active. For more information about these trace types, see "set_cs_trace" on page 547.

If no tracing is active or if tracing is active for all types of messages, one of the following values is returned:

NONE Tracing for all types of messages is inactive.

ALL Tracing for all types of messages is active.

If tracing is active on specific interfaces, one or more of the following values is returned (combined using a + character):

CS_ADMIN_MSG
Internal messages relating to client/server topology are traced.

CS_DATAGRAM
Datagram messages are traced.

CS_DATA
Data messages are traced.

trace_direction
Specifies the direction or directions in which tracing is active. This parameter is not returned if *trace_flags* is set to NONE. Possible values are:

CS_SEND
Messages flowing from the target computer to the computer defined by *dest_sys* are traced.

CS_RECEIVE
Messages flowing from the computer defined by *dest_sys* to the target computer are traced.

CS_BOTH
Messages flowing in both directions are traced.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

NAME_NOT_FOUND
The server specified by the *dest_sys* parameter was not valid or was not started.

LOCAL_SYSTEM
The server specified by the *dest_sys* parameter is the same as the target node to which this command was issued.

INVALID_TARGET
The command was issued on a standalone server. This command can only be issued on a client/server system.

State Check
No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions
Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

query_default_pu

The **query_default_pu** command returns information about the default PU, which is defined using **define_default_pu**.

Supplied Parameters

```
[query_default_pu]
```

No parameters are supplied for this command.

Returned Parameters

Parameter name	Type	Length
def_pu_name	character	8
description	character	31
def_pu_sess	character	8

If the command executes successfully, Communications Server for Linux returns the following parameters:

def_pu_name
Name of the most recently defined default PU. This parameter is blank if no default PU has been defined or the default PU has been deleted.

description
A text string describing the default PU, as specified in the definition of the default PU.

def_pu_sess
Name of the PU associated with the currently active default PU session.

This parameter usually contains the same value as the *def_pu_name* parameter. However, if a new default PU has been defined but the session associated with it is not active, Communications Server for Linux continues to use the session associated with the previous default PU until the session associated with the defined default PU becomes active. In this case, *def_pu_sess* specifies the name of the previous default PU, and is different from the *def_pu_name* parameter.

This parameter is blank if there are no active PU sessions.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

query_defaults

The **query_defaults** command enables the user to query the default parameters defined for the node (default parameters are defined using **define_defaults**).

Supplied Parameters

```
[query_defaults]
```

No parameters are supplied for this command.

Returned Parameters

Parameter name	Type	Length
description	character	31
mode_name	character	8
implicit_plu_forbidden	constant	
specific_security_sense_codes	constant	
limited_timeout	decimal	

If the command executes successfully, the following parameters are returned:

description
: A text string describing the default parameters, as specified on the **define_defaults** command.

mode_name
: Name of the default mode. If an application specifies an unrecognized mode name when attempting to start a session, the parameters from this mode are used as a default definition for the unrecognized mode.

  If no default mode name has been specified using the **define_defaults** command, this parameter is blank.

implicit_plu_forbidden
: Indicates whether Communications Server for Linux puts implicit definitions in place for unknown partner LUs. Possible values are:

  YES Communications Server for Linux does not put implicit definitions in place for unknown partner LUs. All partner LUs must be defined explicitly.

NO Communications Server for Linux puts implicit definitions in place for unknown partner LUs.

specific_security_sense_codes
Indicates whether Communications Server for Linux uses specific sense codes on a security authentication or authorization failure. Specific sense codes are only returned to those partner LUs which have reported support for them on the session. Possible values are:

YES Communications Server for Linux uses specific sense codes.

NO Communications Server for Linux does not use specific sense codes.

limited_timeout
Specifies the timeout after which free limited-resource conwinner sessions are deactivated. The range is `0–65,535` seconds.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists additional combinations of primary and secondary return codes that are common to all commands.

query_directory_entry

The **query_directory_entry** command returns information about resources in the directory database. The command can return summary or detailed information about a specific resource or multiple resources, depending on the options used.

When the command is issued to a running node, it returns information about the resources that have been explicitly defined (using **define_directory_entry** or **define_adjacent_len_node**) and the resources that have been located dynamically in the directory database. If the node is not running, only explicitly defined entries are returned.

When the command is issued to an end node, it returns information only about the end node and its resources, but not about other nodes contained in the directory database. The first entry returned is for the end node, followed by its LUs. (No entry is returned for the end node's network node server.)

When the command is issued to a network node, it returns information about multiple network nodes and their associated end nodes and LUs contained in the directory. For each network node, the information returned is in the following order:

1. The network node.

2. The LUs owned by this node.
3. The first end node associated with the network node.
4. The LUs owned by this end node.
5. Any other end nodes associated with the network node, each followed by its LUs.

Supplied Parameters

Parameter name	Type	Length	Default
[query_directory_entry]			
num_entries	decimal	1	
list_options	constant		SUMMARY + LIST_INCLUSIVE
resource_name	character	17	(null string)
resource_type	constant		NONE
parent_name	character	17	(null string)
parent_type	constant		NONE

Supplied parameters are:

num_entries
Maximum number of resources for which data should be returned. You can specify 1 to return data for a specific resource, a number greater than 1 to return data for multiple resources, or 0 (zero) to return data for all resources.

list_options
The level of information required for each entry and the position in the list of resources from which Communications Server for Linux begins to return data. The list is ordered by *parent_name*, then by *resource_name*, and finally by *resource_type*.

Specify the level of information required with one of the following values:

SUMMARY
Summary information only

DETAIL Detailed information

Use a + character to combine this value with one of the following values:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the combination of the *parent_name*, *resource_name*, and *resource_type* parameters

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the combination of the *parent_name*, *resource_name*, and *resource_type* parameters

resource_name
Fully qualified name of the resource for which information is required, or the name to be used as an index into the list of resources. This value is ignored if *list_options* is set to FIRST_IN_LIST. Specify 3–17 type-A characters consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character resource name.

resource_type
Type of resource for which information is required. This value is ignored if *list_options* is set to FIRST_IN_LIST. Possible values are:

ENCP_RESOURCE
End node (EN) or low-entry networking (LEN) node

NNCP_RESOURCE
Network node (NN)

LU_RESOURCE
Logical unit (LU)

WILDCARD_LU_RESOURCE
Wildcard LU name

NONE All resource types

parent_name
Fully qualified resource name of the parent resource. For an LU, the parent resource is the owning control point and for an end node or LEN node it is the network node server. To return only entries belonging to the specified parent, set this parameter to the name of the parent resource and *parent_type* to the type of the resource parent; to return all entries, do not specify either parameter.

Specify 3–17 type-A characters consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character parent name.

parent_type
Resource type of the parent resource. To return only entries belonging to the specified parent, set this parameter to the type of the parent resource and *parent_name* to the name of the parent resource; to return all entries, do not specify either parameter. Possible values are:

ENCP_RESOURCE
Return only entries belonging to LU resources owned by the end node named by the *parent_name* parameter.

NNCP_RESOURCE
Return only entries belonging to LU resources owned by the network node, end node, or LEN node named by the *parent_name* parameter.

NONE Return entries belonging to all parent resource types.

Returned Parameters: Summary Information

Parameter name	Type	Length
resource_name	character	17
resource_type	constant	
description	character	31
real_owning_cp_type	constant	
real_owning_cp_name	character	17

If the command executes successfully and you specified SUMMARY as the *list_options* parameter value, the following parameters are returned:

resource_name
Fully qualified name of the resource.

resource_type
Type of the resource. Possible values are:

ENCP_RESOURCE
End node (EN) or low-entry networking (LEN) node

NNCP_RESOURCE
Network node (NN)

LU_RESOURCE
Logical unit (LU)

WILDCARD_LU_RESOURCE
Wildcard LU name

description
A text string describing the directory entry, as specified in the definition of the directory entry.

real_owning_cp_type
This parameter applies only if the local node is a Network Node or a Branch Network Node; it is not used otherwise.

Specifies whether the real CP that owns the resource identified by this directory entry is the parent resource or another node. This is one of the following:

NONE The real owner is the parent resource.

ENCP_RESOURCE
The real owner is an end node that is not the parent resource. For example, if the resource is owned by an End Node in the domain of a Branch Network Node (BrNN), the directory of this BrNN's Network Node Server includes the BrNN as the parent resource, but the real owning CP is the End Node.

real_owning_cp_name
This parameter applies only if the local node is a Network Node or a Branch Network Node; it is not used otherwise.

If the *real_owning_cp_type* parameter indicates that the real owner of the resource is not the parent, this parameter specifies the fully qualified name of the CP that owns the resource; otherwise it is not used.

The name consists of a 1–8 character network name, followed by a period, followed by a 1–8 character CP name.

Returned Parameters: Detailed Information

Parameter name	Type	Length
resource_name	character	17
resource_type	constant	
description	character	31
parent_name	character	17
parent_type	constant	
entry_type	constant	
location	constant	
real_owning_cp_type	constant	
real_owning_cp_name	character	17
supplier_cp_type	constant	
supplier_cp_name	character	17

If the command executes successfully and you specified `DETAIL` as the *list_options* parameter value, the following parameters are returned:

resource_name
Fully qualified name of the resource

resource_type
Type of the resource. Possible values are:

ENCP_RESOURCE
End node (EN) or low-entry networking (LEN) node

NNCP_RESOURCE
Network node (NN)

LU_RESOURCE
Logical unit (LU)

WILDCARD_LU_RESOURCE
Wildcard LU name

description
A text string describing the directory entry, as specified in the definition of the directory entry.

parent_name
Fully qualified resource name of the parent resource. For an LU, the parent resource is the owning control point and for an end node or LEN node it is the network node server. This parameter is not used for a network node resource.

parent_type
Resource type of the parent resource. Possible values are:

ENCP_RESOURCE
End node (for an LU resource owned by an EN)

NNCP_RESOURCE
Network node (for an LU resource owned by a NN, or for an EN or LEN resource)

NONE No parent (for a Network Node resource)

entry_type
Specifies the type of the directory entry. Possible values are:

HOME Local resource

CACHE Cached entry

REGISTER
Registered resource (NN only)

location
Specifies the location of the resource. Possible values are:

LOCAL The resource is at the local node.

DOMAIN The resource belongs to an attached end node.

CROSS_DOMAIN
The resource is not within the domain of the local node.

real_owning_cp_type
This parameter applies only if the local node is a Network Node or a Branch Network Node; it is not used otherwise.

Specifies whether the real CP that owns the resource identified by this directory entry is the parent resource or another node. This is one of the following:

NONE The real owner is the parent resource.

ENCP_RESOURCE
The real owner is an end node that is not the parent resource. For

example, if the resource is owned by an End Node in the domain of a Branch Network Node (BrNN), the directory of this BrNN's Network Node Server includes the BrNN as the parent resource, but the real owning CP is the End Node.

real_owning_cp_name
This parameter applies only if the local node is a Network Node or a Branch Network Node; it is not used otherwise.

If the *real_owning_cp_type* parameter indicates that the real owner of the resource is not the parent, this parameter specifies the fully qualified name of the CP that owns the resource; otherwise it is not used.

The name consists of a 1–8 character network name, followed by a period, followed by a 1–8 character CP name.

supplier_cp_type
This parameter applies only if the local node is a Network Node or a Branch Network Node; it is not used otherwise.

Specifies whether this directory entry was registered by another node that is not the owning CP of the resource. This is one of the following:

NONE The directory entry was not registered, or was registered by its owning CP.

ENCP_RESOURCE
The directory entry was registered by a node that is not its owning CP. For example, if the resource is owned by an End Node in the domain of a Branch Network Node (BrNN) that is itself in the domain of the local node, the BrNN is the supplier because it registers the resource with the local node, but the real owning CP is the End Node.

supplier_cp_name
This parameter applies only if the local node is a Network Node or a Branch Network Node; it is reserved otherwise.

If the *supplier_cp_type* parameter indicates that the directory entry was registered by a node that is not the owning resource, this parameter specifies the fully qualified name of the CP that supplied the registration; otherwise it is not used.

The name consists of a 1–8 character network name, followed by a period, followed by a 1–8 character CP name.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_RES_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *resource_name* parameter value was not valid.

INVALID_RES_TYPE
The *resouce_type* parameter was not set to a valid value.

State Check
No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions
Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

query_directory_lu

The **query_directory_lu** command returns a list of LUs from the directory database. The command can be used to obtain information about a specific LU or about multiple LUs, depending on the options used.

This command must be issued to a running node.

Supplied Parameters

Parameter name [query_directory_lu]	Type	Length	Default
num_entries	decimal	1	
list_options	constant		SUMMARY + INCLUSIVE
lu_name	character	17	(null string)

Supplied parameters are:

num_entries
Maximum number of LUs for which data should be returned. You can specify 1 to return data for a specific LU, a number greater than 1 to return data for multiple LUs, or 0 (zero) to return data for all LUs.

list_options
The level of information required for each entry and the position in the list of LUs from which Communications Server for Linux begins to return data.

Specify the level of information required with one of the following values:

SUMMARY
Summary information only

DETAIL Detailed information

Use a + character to combine this value with one of the following values:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the *lu_name* parameter

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *lu_name* parameter

lu_name
Fully qualified name of the LU for which information is required, or the name to be used as an index into the list of LUs. This value is ignored if *list_options* is set to FIRST_IN_LIST. Specify 3–17 type-A characters consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character LU name.

Returned Parameters: Summary Information

Parameter name	Type	Length
lu_name	character	17
description	character	31

If the command executes successfully and you specified SUMMARY as the *list_options* parameter value, the following parameters are returned:

lu_name
Fully qualified name of the LU.

description
A text string describing the directory entry, as specified in the definition of the directory entry.

Returned Parameters: Detailed Information

Parameter name	Type	Length
lu_name	character	17
description	character	31
server_name	character	17
lu_owner_name	character	17
location	constant	
entry_type	constant	
wild_card	constant	
apparent_lu_owner_name	character	17

If the command executes successfully and you specified DETAIL as the *list_options* parameter value, the following parameters are returned:

lu_name
Fully qualified name of the LU.

description
A text string describing the directory entry, as specified in the definition of the directory entry.

server_name
Fully qualified name of the node that serves the LU.

lu_owner_name
Fully qualified name of the node that owns the LU.

location
Specifies the location of the resource. Possible values are:

LOCAL The resource is at the local node.

DOMAIN The resource belongs to an attached end node.

CROSS_DOMAIN
The resource is not within the domain of the local node.

entry_type
Specifies the type of the resource. Possible values are:

HOME Local resource

CACHE Cached entry

REGISTER
Registered resource (NN only)

wild_card
Specifies whether the LU entry is for an explicit name or for a wildcard value that will match a range of names. Possible values are:

EXPLICIT
The entry is an explicit LU name.

FULL_WILDCARD
The entry is a full wildcard value that will match any LU name.

PARTIAL_WILDCARD
The entry is a partial wildcard; the nonblank characters in the name will be used to match an LU name.

OTHER The entry is an unknown type.

apparent_lu_owner_name
This parameter applies only if the local node is a Network Node or a Branch Network Node; it is not used otherwise.

If the apparent owning CP of this LU is not the real owning CP, this parameter specifies the fully qualified name of the apparent owning CP; otherwise it is not used. For example, if the resource is owned by an End Node in the domain of a Branch Network Node (BrNN), the directory of this BrNN's Network Node Server includes the BrNN as the apparent owner, but the real owning CP is the End Node.

The name consists of a 1–8 character network name, followed by a period, followed by a 1–8 character CP name.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_LU_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *lu_name* parameter value was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

query_directory_stats

The **query_directory_stats** command returns directory database statistics, that can be used to gauge the level of network locate traffic. For a network node, it returns statistics on the usage of the directory cache; you can use this information to determine the appropriate cache size, which is specified in the node definition.

This command must be issued to a running node.

Supplied Parameters

```
[query_directory_stats]
```

No parameters are supplied for this command.

Returned Parameters

```
Parameter name            Type
max_caches                decimal
cur_caches                decimal
cur_home_entries          decimal
cur_reg_entries           decimal
cur_directory_entries     decimal
cache_hits                decimal
cache_misses              decimal
in_locates                decimal
in_bcast_locates          decimal
out_locates               decimal
out_bcast_locates         decimal
not_found_locates         decimal
not_found_bcast_locates   decimal
locates_outstanding       decimal
```

If the command executes successfully, the following parameters are returned:

max_caches
: For a network node, the maximum number of cache entries allowed.

cur_caches
: For a network node, the current number of cache entries.

cur_home_entries
: Current number of home entries.

cur_reg_entries
: Current number of registered entries.

cur_directory_entries
: Total number of entries currently in the directory.

cache_hits
: For a network node, the number of successful cache finds. The count is increased every time a resource is found in the local directory cache.

cache_misses
: For a network node, the number of times a resource has been found by a broadcast search. The count is increased every time a resource is not found in the local directory cache but is then found using a broadcast search.

Note: The two counts *cache_hits* and *cache_misses* are maintained such that the size of the directory cache (specified on **define_node**) can be tuned. An increasing *cache_misses* over time indicates that the directory cache size is too small. A regularly increasing *cache_hits* with a steady *cache_misses* indicates that the cache is about the right size.

in_locates
Number of directed locates received.

in_bcast_locates
For a network node, the number of broadcast locates received.

out_locates
Number of directed locates sent.

out_bcast_locates
For a network node, the number of broadcast locates sent.

not_found_locates
Number of directed locates returned "not found."

not_found_bcast_locates
For a network node, the number of broadcast locates returned "not found."

locates_outstanding
Current number of outstanding locates, both directed and broadcast.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_dlc

The **query_dlc** command returns information about DLCs. The command can be used to obtain summary or detailed information about a specific DLC or about multiple DLCs, depending on the options used.

Supplied Parameters

Parameter name	Type	Length	Default
[query_dlc]			
num_entries	decimal		1
list_options	constant		SUMMARY + LIST_INCLUSIVE
dlc_name	character	8	(null string)

Supplied parameters are:

num_entries
Maximum number of DLCs for which data should be returned. You can specify 1 to return data for a specific DLC, a number greater than 1 to return data for multiple DLCs, or 0 (zero) to return data for all DLCs.

list_options
The level of information required for each entry and the position in the list of DLCs from which Communications Server for Linux begins to return data.

Specify the level of information required with one of the following values:

SUMMARY
Summary information only

DETAIL Detailed information

Use a + character to combine this value with one of the following values:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the *dlc_name* parameter

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *dlc_name* parameter

dlc_name
Name of the DLC for which information is required, or the name to be used as an index into the list of DLCs. This parameter is ignored if *list_options* is set to FIRST_IN_LIST.

Returned Parameters: Summary Information

Parameter name	Type	Length
dlc_name	character	8
description	character	31
state	constant	
dlc_type	constant	

If the command executes successfully and you specified SUMMARY as the *list_options* parameter value, Communications Server for Linux returns the following parameters:

dlc_name
Name of the DLC.

description
A text string describing the DLC, as specified in the definition of the DLC.

state State of the DLC. Possible values are:

ACTIVE The DLC is active.

NOT_ACTIVE
The DLC is not active.

PENDING_INACTIVE
The **stop_dlc** command is in progress.

dlc_type
Type of the DLC. Possible values are:

SDLC Synchronous data link control

QLLC Qualified logical link control

TR Token Ring

ETHERNET
Ethernet

MPC Multipath Channel (MPC) (Communications Server for Linux on System z only)

HPRIP Enterprise Extender (HPR/IP)

Returned Parameters: Detailed Information

Parameter name	Type	Length
dlc_name	character	8
description	character	31
state	constant	
dlc_type	constant	
initially_active	constant	

For SDLC, the following parameters are included. For information about these parameters, see "define_sdlc_dlc" on page 161.

neg_ls_supp	constant
adapter_number	decimal
mu_credit	decimal
stats_support	constant
num_ports	decimal
creators_pid	decimal

For QLLC, the following parameters are included. For information about these parameters, see "define_qllc_dlc" on page 134.

adapter_number	decimal

For Ethernet, the following parameters are included. For information about these parameters, see "define_tr_dlc, define_ethernet_dlc" on page 207.

neg_ls_supp	constant
adapter_number	decimal
lan_type	constant

For Token Ring, the following parameters are included. For information about these parameters, see "define_tr_dlc, define_ethernet_dlc" on page 207.

neg_ls_supp	constant
adapter_number	decimal

For Multipath Channel (MPC), Communications Server for Linux on System z only, the following parameter is included.

stats_support	decimal

For Enterprise Extender (HPR/IP), the following additional parameters are included. For information about these parameters, see "define_ip_dlc" on page 65.

udp_port_llc	decimal
udp_port_network	decimal
udp_port_high	decimal
udp_port_medium	decimal
udp_port_low	decimal
ip_precedence_llc	decimal
ip_precedence_network	decimal
ip_precedence_high	decimal
ip_precedence_medium	decimal
ip_precedence_low	decimal
no_dns_lookup	constant

If the command executes successfully and you specified DETAIL as the *list_options* parameter value, Communications Server for Linux returns the following parameters:

dlc_name
DLC name.

description
A text string describing the DLC, as specified in the definition of the DLC.

state State of the DLC. Possible values are:

ACTIVE The DLC is active.

NOT_ACTIVE
The DLC is not active.

PENDING_INACTIVE
The **stop_dlc** command is in progress.

dlc_type
Type of the DLC. Possible values are:

SDLC Synchronous data link control

QLLC Qualified link level control

TR Token Ring

ETHERNET
Ethernet

MPC Multipath Channel (MPC) (Communications Server for Linux on System z only)

HPRIP Enterprise Extender (HPR/IP)

initially_active
Indicates whether this DLC is automatically started when the node is started. Possible values are:

YES The DLC is automatically started when the node is started.

NO The DLC is not automatically started; it must be manually started.

For Multipath Channel (MPC), Communications Server for Linux on System z only, the following parameter is included.

stats_support
Statistics support. This parameter is set to NO to indicate that statistics information is not available for the DLC.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible value is:

INVALID_DLC_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the value specified in the *dlc_name* parameter was not valid.

State Check
No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions
Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

query_dlc_trace

The **query_dlc_trace** command returns information about DLC line tracing, which was defined using **add_dlc_trace** commands. This command can be used to obtain information about tracing on all resources, on a specific resource type, or on a specific resource, depending on the options used.

Supplied Parameters

Parameter name	Type	Length	Default
[query_dlc_trace]			
num_entries	decimal		1
list_options	constant		LIST_INCLUSIVE
list_type	constant		ALL_DLC_TRACES
resource_type	constant		ALL_RESOURCES
resource_name	character	8	(null string)
sidh	hex byte		0
sidl	hex byte		0
odai	constant		NO

Supplied parameters are:

num_entries
Maximum number of entries for which data should be returned. You can specify 1 to return data for a specific entry, a number greater than 1 to return data for multiple entries, or 0 (zero) to return data for all entries.

list_options
The position in the list of entries from which Communications Server for Linux begins to return data. The list is ordered by *resource_type* and then by *resource_name*.

Possible values are:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the *resource_type* and *resource_name* parameters

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *resource_type* and *resource_name* parameters

list_type
The type of resource for which to list tracing options. Possible values are:

ALL_DLC_TRACES
List all specified tracing options (for any resource type).

ALL_RESOURCES
List the tracing options specified for all resources (defined using **add_dlc_trace** with a resource type of ALL_RESOURCES).

DLC List tracing options for DLC resources.

PORT List tracing options for port resources for which all link stations are traced.

LS List tracing options for link station (LS) resources.

RTP List tracing options for RTP connection resources.

PORT_DEFINED_LS
List tracing options for port resources for which only defined link stations (not implicit link stations) are traced.

PORT_IMPLICIT_LS
List tracing options for port resources for which only implicit link stations (not defined link stations) are traced.

resource_type
Specifies the resource type of the entry to be returned, or the entry to be used as an index into the list. This parameter is used only if *list_type* is set to ALL_DLC_TRACES and *list_options* is not set to FIRST_IN_LIST. Possible values are:

ALL_RESOURCES
The required entry specifies the options used for tracing all DLCs, ports, link stations, and RTP connections.

DLC The required entry specifies tracing options for the DLC named in *resource_name* and for all ports and link stations that use this DLC.

PORT The required entry specifies tracing options for the port named in *resource_name* and for all link stations that use this port.

LS The required entry specifies tracing options for the LS named in *resource_name*.

RTP The required entry specifies tracing options for the RTP connection named in the *resource_name* parameter.

PORT_DEFINED_LS
The required entry specifies tracing options for the port named in *resource_name* and for all defined link stations (but not implicit link stations) that use this port.

PORT_IMPLICIT_LS
The required entry specifies tracing options for the port named in *resource_name* and for all implicit link stations (but not defined link stations) that use this port.

resource_name
The name of the entry to be returned, or the entry to be used as an index into the list. This parameter is ignored if *list_options* is set to FIRST_IN_LIST, or if *resource_type* is set to ALL_RESOURCES.

The following three parameters identify the Local Form Session Identifier (LFSID) for a session on the specified LS. This LFSID is valid only if *resource_type* is set to LS and indicates that only messages on this session are to be traced.

The LFSID consists of the following parameters:

sidh Session ID high byte

sidl Session ID low byte

odai Origin Destination Assignor Indicator. Possible values are:

YES The BIND sender is the node containing the secondary link station.

NO The BIND sender is the node containing the primary link station.

Returned Parameters

Parameter name	Type	Length
resource_type	constant	
resource_name	character	8
sidh	hex byte	
sidl	hex byte	
odai	constant	
message_type	constant	

If the command executes successfully, Communications Server for Linux returns the following parameters:

resource_type
The type of resource to be traced. Possible values are:

ALL_RESOURCES
The entry specifies tracing options for all resources.

DLC The entry specifies tracing options for the DLC named in *resource_name* and for all ports and link stations that use this DLC.

PORT The entry specifies tracing options for the port named in *resource_name* and for all link stations that use this port.

LS The entry specifies tracing options for the LS named in *resource_name* (or for a particular LFSID on this LS).

RTP The entry specifies tracing options for the RTP connection named in *resource_name*.

PORT_DEFINED_LS
The entry specifies tracing options for the port named in *resource_name* and for all defined link stations (but not implicit link stations) that use this port.

PORT_IMPLICIT_LS
The entry specifies tracing options for the port named in *resource_name* and for all implicit link stations (but not defined link stations) that use this port.

resource_name
The name of the DLC, port, LS, or RTP connection to be traced.

The following three parameters identify the Local Form Session Identifier for a session on the specified LS. This LFSID is valid only if *resource_type* is set to LS and indicates that only messages on this session are to be traced. The LFSID consists of the following parameters:

sidh Session ID high byte

sidl Session ID low byte

odai Origin Destination Assignor Indicator. Possible values are:

YES The BIND sender is the node containing the secondary link station.

NO The BIND sender is the node containing the primary link station.

message_type
The type of messages to be traced for the specified resource or session. If all messages are being traced, this parameter is set to TRACE_ALL. If specific messages are being traced, one or more of the following values is combined with a + character:

TRACE_XID
XID messages

TRACE_SC
Session control RUs

TRACE_DFC
Data flow control RUs

TRACE_FMD
FMD messages

TRACE_NLP
Network layer protocol messages

TRACE_NC
Network control messages

TRACE_SEGS
Non-BBIU segments that do not contain an RH

TRACE_CTL
Messages other than MUs and XIDs

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_LIST_TYPE
The value specified in the *list_type* parameter was not valid.

INVALID_RESOURCE_TYPE
The value specified in the *resource_type* parameter was not valid.

ALL_RESOURCES_NOT_DEFINED
The *resource_type* parameter was set to ALL_RESOURCES, but a DLC_TRACE entry has not been defined for tracing options on all resources.

INVALID_RTP_CONNECTION
The RTP connection named in the *resource_name* parameter does not have any tracing options set.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_dlur_defaults

The **query_dlur_defaults** command allows the user to query the defaults that were defined using the **define_dlur_defaults** command.

Supplied Parameters

```
[query_dlur_defaults]
```

No parameters are supplied for this command.

Returned Parameters

Parameter name	Type	Length
description	character	31
dlus_name	character	17
bkup_dlus_name	character	17
dlus_retry_timeout	decimal	
dlus_retry_limit	decimal	

If the command executes successfully the following parameters are returned:

description
A text string describing the DLUR defaults.

dlus_name
Name of the DLUS node that is the default.

bkup_dlus_name
Name of the DLUS node that serves as the backup default.

dlus_retry_timeout
Reactivation timer for contacting a DLUS. If Communications Server for Linux fails to contact the DLUS, this parameter indicates the time in seconds between retries.

dlus_retry_limit
Retry count for contacting a DLUS. The value of this parameter indicates the number of times Communications Server for Linux retries if it fails to contact the DLUS on the first attempt.

A value of 65,535 indicates that Communications Server for Linux retries indefinitely until it contacts the DLUS.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Function Not Supported

If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameters:

primary_rc

FUNCTION_NOT_SUPPORTED
The node does not support DLUR; support is defined by the *dlur_support* parameter on the **define_node** command.

secondary_rc
(This parameter is not used.)

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

query_dlur_lu

The **query_dlur_lu** command returns information for active LUs that are using the DLUR feature of Communications Server for Linux. This command can be used to obtain information for a specific LU or about multiple LUs, depending on the options used.

This command must be issued to a running node.

Supplied Parameters

Parameter name	Type	Length	Default
[query_dlur_lu]			
num_entries	decimal		1
list_options	constant		SUMMARY + LIST_INCLUSIVE
lu_name	character	8	(null string)
pu_name	character	8	(null string)
filter	constant		NONE

Supplied parameters are:

num_entries
Maximum number of DLUR LUs for which data should be returned. You can specify 1 to return data for a specific DLUR LU, a number greater than 1 to return data for multiple DLUR LUs, or `0` (zero) to return data for all DLUR LUs.

list_options
The level of information required for each entry and the position in the list

of DLUR LUs from which Communications Server for Linux begins to return data. The list is ordered by *pu_name* and then by *lu_name*.

Specify the level of information required with one of the following values:

SUMMARY
Summary information only

DETAIL Detailed information

Use a + character to combine this value with one of the following values:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the combination of the *pu_name* and *lu_name* parameters

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the combination of the *pu_name* and *lu_name* parameters

lu_name
Name of the LU for which information is required, or the name to be used as an index into the list of LUs. This value is ignored if *list_options* is set to FIRST_IN_LIST. The name is a type-A character string.

pu_name
Name of the PU for which LU information is required. To list only information about LUs associated with a specific PU, specify the PU name. To obtain a complete list for all PUs, set this parameter to binary zeros. The name is a type-A character string.

filter Specifies whether to filter the returned LUs according to their location. Possible values for network node are:

INTERNAL
Return information only for internal LUs.

DOWNSTREAM
Return information only for downstream LUs.

NONE Return information for all LUs regardless of location.

For end node, this parameter is reserved (downstream DLUR LUs are not supported).

Returned Parameters: Summary Information

Parameter name	Type	Length
lu_name	character	8

If the command executes successfully and you specified SUMMARY as the *list_options* parameter value, Communications Server for Linux returns the following parameters:

lu_name
Name of the LU.

Returned Parameters: Detailed Information

Parameter name	Type	Length
lu_name	character	8
pu_name	character	8

```
dlus_name               character      17
lu_location             constant
nau_address             decimal
plu_name                character      17
rscv_len                hex array      256
```

If the command executes successfully and you specified DETAIL as the *list_options* parameter value, Communications Server for Linux returns the following parameters:

lu_name
Name of the LU.

pu_name
Name of the PU associated with the LU.

dlus_name
If the SSCP-LU session is active, this parameter contains the name of the DLUS node used by the LU; it is not used otherwise.

lu_location
Location of LU.

Possible values are:

INTERNAL
LU is on the local node.

DOWNSTREAM
LU is on a downstream node (network node only).

nau_address
Network accessible unit (NAU) address of the LU, in the range 1–255.

plu_name
If the PLU-SLU session is active, this parameter contains the name of the PLU; otherwise, it is set to 17 zeros.

rscv_len
Route Selection control vector (RSCV) as defined in *Systems Network Architecture: Formats*. This control vector defines the session route through the network and is carried on the BIND. This RSCV is included only if the node's configuration indicates that DLUR RSCVs should be stored and the PLU-SLU session is active.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_LU_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *lu_name* parameter value was not valid.

INVALID_FILTER_OPTION
The *filter* parameter was not set to a valid value.

State Check
No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Function Not Supported
If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameters:

primary_rc

FUNCTION_NOT_SUPPORTED
The local node does not support DLUR; this support is defined by the *dlur_support* parameter on the **define_node** command.

secondary_rc
(This parameter is not used.)

Other Conditions
Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_dlur_pu

The **query_dlur_pu** command returns information about PUs that use the DLUR feature of Communications Server for Linux. This command can be used to obtain information about a specific PU or about multiple PUs, depending on the options used.

If this command is issued to an inactive node, it returns information only about PUs defined at the local node. If this command is issued to a running node, it returns information about PUs defined at the local node and about active downstream PUs using DLUR at this node.

Supplied Parameters

Parameter name [query_dlur_pu]	Type	Length	Default
num_entries	decimal		1
list_options	constant		SUMMARY + LIST_INCLUSIVE
pu_name	character	8	(null string)
dlus_name	character	17	(null string)
filter	constant		NONE

Supplied parameters are:

num_entries
Maximum number of DLUR PUs for which data should be returned. You can specify 1 to return data for a specific DLUR PU, a number greater than 1 to return data for multiple DLUR PUs, or 0 (zero) to return data for all DLUR PUs.

list_options
The level of information required for each entry and the position in the list of DLUR PUs from which Communications Server for Linux begins to return data. The list is ordered by *pu_name*.

Specify the level of information required with one of the following values:

SUMMARY
Summary information only

DETAIL Detailed information

Use a + character to combine this value with one of the following values:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the *pu_name* parameter

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *pu_name* parameter

pu_name
Name of the PU for which information is required, or the name to be used as an index into the list of PUs. This value is ignored if *list_options* is set to FIRST_IN_LIST. The name is a type-A character string.

dlus_name
Name of the DLUS for which PU information is required. To list only information about PUs associated with a specific DLUS, specify the DLUS name; a PU will be listed only if it has an SSCP-PU session to the specified DLUS node. To obtain a complete list for all DLUSs, do not specify this parameter.

Specify 3–17 type-A characters consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character DLUS name.

filter Specifies whether to filter the returned PUs according to their location. Possible values for network node are:

INTERNAL
Return information only about internal PUs.

DOWNSTREAM
Return information only about downstream PUs.

NONE Return information about all PUs regardless of location.

For end node, this parameter is reserved (downstream DLUR PUs are not supported).

Returned Parameters: Summary Information

Parameter name	Type	Length
pu_name	character	8
description	character	31

If the command executes successfully and you specified SUMMARY as the *list_options* parameter value, Communications Server for Linux returns the following parameters:

pu_name
Name of the PU.

description
A text string describing the PU, as specified in the definition of the PU. This parameter is reserved if the PU is an active downstream PU, rather than a defined internal PU.

Returned Parameters: Detailed Information

Parameter name	Type	Length
pu_name	character	8
description	character	31
defined_dlus_name	character	17
bkup_dlus_name	character	17
pu_id	hex array	4
pu_location	constant	
active_dlus_name	character	17
ans_support	constant	
pu_status	constant	
dlus_session_status	constant	
pcid	hex array	8
fqcp_name	character	17
initially_active	constant	
dlus_retry_timeout		
dlus_retry_limit		

If the command executes successfully and you specified DETAIL as the *list_options* parameter value, Communications Server for Linux returns the following parameters:

pu_name
Name of the PU.

description
A text string describing the PU, as specified in the definition of the PU. This parameter is reserved if the PU is an active downstream PU, rather than a defined internal PU.

defined_dlus_name
Name of the DLUS node, defined by either the **define_internal_pu** or **define_*_ls** command (with *dspu_services* set to DLUR).

bkup_dlus_name
Name of the backup DLUS node, defined by either the **define_internal_pu** or **define_*_ls** command (with *dspu_services* set to DLUR).

pu_id PU identifier, either defined on **define_internal_pu** or obtained in an XID from a downstream PU. This ID is a 4-byte hexadecimal string, consisting of a block number (three hexadecimal digits) and a node number (five hexadecimal digits).

pu_location
Location of the PU.

Possible values are:

INTERNAL
The PU is on the local node.

DOWNSTREAM
The PU is on a downstream node (network node only).

active_dlus_name
Name of the DLUS node that the PU is currently using. If the SSCP-PU session is not active, this parameter is not returned.

ans_support
Auto network shutdown (ANS) support, defined by the DLUS and sent to DLUR from the DLUS at SSCP-PU activation. This parameter specifies whether to continue link-level contact if the subarea node initiates an auto network shutdown procedure for the SSCP controlling the PU. Possible values are:

CONT Continue link-level contact.

STOP Stop link-level contact.

This parameter is reserved if the SSCP-LU session is inactive.

pu_status
Status of the PU (associated with DLUR). Possible values are:

RESET The PU is in reset state.

PEND_ACTPU
The PU is waiting for an ACTPU from the host.

PEND_ACTPU_RSP
DLUR is waiting for the PU to respond to a forwarded ACTPU.

ACTIVE The PU is active.

PEND_DACTPU_RSP
DLUR is waiting for the PU to respond to a forwarded DACTPU.

PEND_INOP
DLUR is waiting for all necessary events to be completed before it deactivates the PU.

dlus_session_status
Status of the DLUS pipe currently being used by the PU. Possible values are:

PENDING_ACTIVE
The pipe is being activated.

ACTIVE The pipe is active.

PENDING_INACTIVE
The pipe is being deactivated.

INACTIVE
The pipe is not active.

pcid Procedure correlator ID (PCID) used on the DLUS pipe. If the SSCP-PU session is not active, this parameter is not used.

fqcp_name
Fully qualified control point name used on the DLUS pipe. If the SSCP-PU session is not active, this parameter is not used.

The combination of the *pcid* and *fqcp_name* parameters uniquely identifies each PU whose sessions are being routed using DLUR. The *fqcp_name* parameter is the CP name of either the DLUR or DLUS node, depending on which node initiated the SSCP-PU session activation.

initially_active
Specifies whether this PU is automatically started when the node is started. For a downstream PU, this parameter is not used. Possible values for an internal PU are:

YES The PU is automatically started when the node is started.

NO The PU is not automatically started; it must be manually started.

dlus_retry_timeout
Time interval in seconds between attempts to contact the DLUS and backup DLUS. A value of 0 (zero) indicates that the value from the **define_dlur_defaults** command is used.

dlus_retry_limit
Number of attempts to recontact a DLUS after an initial failure. A value of 0 (zero) indicates that the value from the **define_dlur_defaults** command is used.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_PU_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *pu_name* parameter value was not valid.

INVALID_FILTER_OPTION
The *filter* parameter was not set to a valid value.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Function Not Supported

If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameters:

primary_rc

FUNCTION_NOT_SUPPORTED
The local node does not support DLUR; this support is defined by the *dlur_support* parameter on the **define_node** command.

secondary_rc
(This parameter is not used.)

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_dlus

The **query_dlus** command returns information about dependent LU server (DLUS) nodes known to the dependent LU requester (DLUR) feature of Communications Server for Linux. This command can be used to obtain information about a specific DLUS or about multiple DLUSs, depending on the options used. This command also returns pipe statistics (SSCP-PU and SSCP-LU session statistics); the **query_isr_session** command can be used to obtain PLU-SLU session statistics.

If this command is issued to an inactive node, it returns information only about DLUS nodes defined using **define_internal_pu** or **define_dlur_defaults**. If this command is issued to a running node, it returns information about DLUS nodes defined using **define_internal_pu** or **define_dlur_defaults** and active DLUS nodes. The **query_dlus** command does not return information about the backup DLUS that was defined using **define_dlur_defaults**, unless the backup DLUS is active.

Supplied Parameters

Parameter name	Type	Length	Default
[query_dlus]			
num_entries	decimal		1
list_options	constant		LIST_INCLUSIVE
dlus_name	character	17	(null string)

Supplied parameters are:

num_entries
: Maximum number of DLUSs for which data should be returned. You can specify 1 to return data for a specific DLUS, a number greater than 1 to return data for multiple DLUSs, or 0 (zero) to return data for all DLUSs.

list_options
: The position in the list of DLUSs from which Communications Server for Linux begins to return data. The list is ordered by *dlus_name*.

  Possible values are:

  FIRST_IN_LIST
  : Start at the first entry in the list

  LIST_INCLUSIVE
  : Start at the entry specified by the *dlus_name* parameter

  LIST_FROM_NEXT
  : Start at the entry immediately following the entry specified by the *dlus_name* parameter

dlus_name
: Name of the DLUS for which information is required, or the name to be used as an index into the list of DLUSs. This value is ignored if *list_options* is set to FIRST_IN_LIST. Specify 3–17 type-A characters consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character DLUS name.

Returned Parameters

Parameter name	Type	Length
dlus_name	character	17
is_default	constant	
is_backup_default	constant	
pipe_state	constant	
num_active_pus	decimal	

```
reqactpu_sent            decimal
reqactpu_rsp_received    decimal
actpu_received           decimal
actpu_rsp_sent           decimal
reqdactpu_sent           decimal
reqdactpu_rsp_received   decimal
dactpu_received          decimal
dactpu_rsp_sent          decimal
actlu_received           decimal
actlu_rsp_sent           decimal
dactlu_received          decimal
dactlu_rsp_sent          decimal
sscp_pu_mus_rcvd         decimal
sscp_pu_mus_sent         decimal
sscp_lu_mus_rcvd         decimal
sscp_lu_mus_sent         decimal
```

If the command executes successfully, Communications Server for Linux returns the following parameters:

dlus_name
Name of the DLUS.

is_default
Specifies whether the DLUS node has been designated as the default by a **define_dlur_defaults** command. Possible values are:

YES DLUS node has been designated as the default.

NO DLUS node has not been designated as the default.

is_backup_default
Specifies whether the DLUS node has been designated as the backup default by a **define_dlur_defaults** command. Possible values are:

YES DLUS node has been designated as the backup default.

NO DLUS node has not been designated as the backup default.

pipe_state
State of the pipe to the DLUS. Possible values are:

PENDING_ACTIVE
The pipe is being activated.

ACTIVE The pipe is active.

PENDING_INACTIVE
The pipe is being deactivated.

INACTIVE
The pipe is not active.

num_active_pus
Number of PUs currently using the pipe to the DLUS.

reqactpu_sent
Number of REQACTPUs sent to DLUS over the pipe to request the activation of a PU.

reqactpu_rsp_received
Number of RSP(REQACTPU)s received from DLUS over the pipe.

actpu_received
Number of ACTPUs received from DLUS over the pipe to activate a PU.

actpu_rsp_sent
Number of RSP(ACTPU)s sent to DLUS over the pipe.

reqdactpu_sent
Number of REQDACTPUs sent to DLUS over the pipe to request the deactivation of a PU.

reqdactpu_rsp_received
Number of RSP(REQDACTPU)s received from DLUS over the pipe.

dactpu_received
Number of DACTPUs received from DLUS over the pipe to deactivate a PU.

dactpu_rsp_sent
Number of RSP(DACTPU)s sent to DLUS over the pipe.

actlu_received
Number of ACTLUs received from DLUS over the pipe to activate an LU.

actlu_rsp_sent
Number of RSP(ACTLU)s sent to DLUS over the pipe.

dactlu_received
Number of DACTLUs received from DLUS over the pipe to deactivate an LU.

dactlu_rsp_sent
Number of RSP(DACTLU)s sent to DLUS over the pipe.

sscp_pu_mus_rcvd
Number of SSCP-PU message units (MUs) received from DLUS over the pipe.

sscp_pu_mus_sent
Number of SSCP-PU message units (MUs) sent to DLUS over the pipe.

sscp_lu_mus_rcvd
Number of SSCP-LU message units (MUs) received from DLUS over the pipe.

sscp_lu_mus_sent
Number of SSCP-LU message units (MUs) sent to DLUS over the pipe.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
`PARAMETER_CHECK`

secondary_rc
Possible values are:

`INVALID_DLUS_NAME`
The *list_options* parameter was set to `LIST_INCLUSIVE` to list all entries starting from the supplied name, but the *dlus_name* parameter value was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Function Not Supported

If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameters:

primary_rc

FUNCTION_NOT_SUPPORTED
The local node does not support DLUR; this support is defined by the *dlur_support* parameter on the **define_node** command.

secondary_rc
(This parameter is not used.)

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_domain_config_file

The **query_domain_config_file** command returns the header information included in the domain configuration file (the Communications Server for Linux version number, the revision level of the file, and an optional comment string).

This command must be issued without specifying a node name.

Supplied Parameters

```
[query_domain_config_file]
```

No parameters are supplied for this command.

Returned Parameters

Parameter name	Type	Length
major_version	decimal	
minor_version	decimal	
update_release	decimal	
revision_level	decimal	
comment	character	99

If the command executes successfully, Communications Server for Linux returns the following parameters:

major_version **through** *update_release*
The internal version identifier of the release of Communications Server for Linux that was used to create this file.

revision_level
The revision level of the file (stored internally by Communications Server for Linux).

comment
An optional comment string containing information about the file, specified on the **define_domain_config_file** command.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

query_downstream_lu

The **query_downstream_lu** command returns information about downstream LUs that use SNA gateway and DLUR. This information is structured as determined data (data gathered dynamically during execution and returned only if the node is active) and defined data (data supplied on **define_downstream_lu**). For DLUR-supported LUs, implicitly defined data is put in place when the downstream LU is activated.

This command can be used to obtain information about a specific LU or about multiple LUs, depending on the options used.

Supplied Parameters

Parameter name [query_downstream_lu]	Type	Length	Default
num_entries	decimal		1
list_options	constant		SUMMARY + LIST_INCLUSIVE
dslu_name	character	8	(null string)
dspu_name	character	8	(null string)
dspu_services	constant		NONE

Supplied parameters are:

num_entries
Maximum number of downstream LUs for which data should be returned. You can specify 1 to return data for a specific downstream LU, a number greater than 1 to return data for multiple downstream LUs, or 0 (zero) to return data for all downstream LUs.

list_options
The level of information required for each entry and the position in the list from which Communications Server for Linux begins to return data. The list is ordered by *dspu_name* and then by *dslu_name*.

Specify the level of information required with one of the following values:

SUMMARY
Summary information only

DETAIL Detailed information

Use a + character to combine this value with one of the following values:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the combination of the *dspu_name* and *dslu_name* parameters

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the combination of the *dspu_name* and *dslu_name* parameters

dslu_name
Name of the LU for which information is required, or the name to be used as an index into the list of LUs. This value is ignored if *list_options* is set to FIRST_IN_LIST. The name is a type-A character string.

dspu_name
PU name for which LU information is required, as specified in the definition of an LS. To list only information about LUs associated with a specific PU, specify the PU name. To obtain a complete list for all PUs, do not specify this parameter.

dspu_services
DSPU services filter. When the **query_downstream_lu** command is issued to a running node, this parameter specifies whether to filter the returned information by the type of services provided to the LUs. Possible values are:

PU_CONCENTRATION
Return information only about downstream LUs served by SNA gateway.

DLUR Return information only about downstream LUs served by DLUR.

NONE Return information about all downstream LUs.

When the node is not running, this parameter is ignored; Communications Server for Linux returns information about all downstream LUs.

Returned Parameters: Summary Information

Parameter name	Type	Length
dslu_name	character	8
dspu_name	character	8
description	character	31
dspu_services	constant	
nau_address	decimal	
lu_sscp_sess_active	constant	
plu_sess_active	constant	

If the command executes successfully and you specified SUMMARY as the *list_options* parameter value, Communications Server for Linux returns the following parameters:

dslu_name
Name of the downstream LU.

dspu_name
Name of the PU associated with the downstream LU.

description
For an LU supported by SNA gateway, this parameter is a text string describing the downstream LU, as specified in the definition of the downstream LU. For a DLUR-supported LU, this parameter is reserved.

dspu_services
When the **query_downstream_lu** command is issued to a running node, this parameter specifies the services provided by the local node to the downstream LU.

Possible values are:

PU_CONCENTRATION
Downstream LU is served by SNA gateway.

DLUR Downstream LU is served by DLUR.

nau_address
Network accessible unit (NAU) address of the downstream LU. This address is in the range 1–255.

lu_sscp_sess_active
Specifies whether the LU-SSCP session is active. Possible values are:

YES The session is active.

NO The session is not active.

plu_sess_active
Specifies whether the PLU-SLU session is active. Possible values are:

YES The session is active.

NO The session is not active.

Returned Parameters: Detailed Information

Parameter name	Type	Length
dslu_name	character	8
lu_sscp_sess_active	constant	
plu_sess_active	constant	
dspu_services	constant	
lu_sscp_rcv_ru_size	decimal	
lu_sscp_send_ru_size	decimal	
lu_sscp_max_send_btu_size	decimal	
lu_sscp_max_rcv_btu_size	decimal	
lu_sscp_max_send_pac_win	decimal	
lu_sscp_cur_send_pac_win	decimal	
lu_sscp_max_rcv_pac_win	decimal	
lu_sscp_cur_rcv_pac_win	decimal	
lu_sscp_send_data_frames	decimal	
lu_sscp_send_fmd_data_frames	decimal	
lu_sscp_send_data_bytes	decimal	
lu_sscp_rcv_data_frames	decimal	
lu_sscp_rcv_fmd_data_frames	decimal	
lu_sscp_rcv_data_bytes	decimal	
lu_sscp_sidh	hex number	
lu_sscp_sidl	hex number	
lu_sscp_odai	constant	
lu_sscp_ls_name	character	8
lu_sscp_pacing_type	constant	
ds_plu_rcv_ru_size	decimal	
ds_plu_send_ru_size	decimal	
ds_plu_max_send_btu_size	decimal	
ds_plu_max_rcv_btu_size	decimal	
ds_plu_max_send_pac_win	decimal	
ds_plu_cur_send_pac_win	decimal	
ds_plu_max_rcv_pac_win	decimal	
ds_plu_cur_rcv_pac_win	decimal	
ds_plu_send_data_frames	decimal	
ds_plu_send_fmd_data_frames	decimal	
ds_plu_send_data_bytes	decimal	

```
ds_plu_rcv_data_frames        decimal
ds_plu_rcv_fmd_data_frames    decimal
ds_plu_rcv_data_bytes         decimal
ds_plu_sidh                   hex number
ds_plu_sidl                   hex number
ds_plu_odai                   constant
ds_plu_ls_name                character     8
ds_plu_pacing_type            constant
us_plu_rcv_ru_size            decimal
us_plu_send_ru_size           decimal
us_plu_max_send_btu_size      decimal
us_plu_max_rcv_btu_size       decimal
us_plu_max_send_pac_win       decimal
us_plu_cur_send_pac_win       decimal
us_plu_max_rcv_pac_win        decimal
us_plu_cur_rcv_pac_win        decimal
us_plu_send_data_frames       decimal
us_plu_send_fmd_data_frames   decimal
us_plu_send_data_bytes        decimal
us_plu_rcv_data_frames        decimal
us_plu_rcv_fmd_data_frames    decimal
us_plu_rcv_data_bytes         decimal
us_plu_sidh                   hex number
us_plu_sidl                   hex number
us_plu_odai                   constant
us_plu_ls_name                character     8
us_plu_pacing_type            constant
description                   character     31
nau_address                   decimal
dspu_name                     character     8
host_lu_name                  character     8
allow_timeout                 constant
delayed_logon                 constant
```

If the command executes successfully and you specified `DETAIL` as the *list_options* parameter value, Communications Server for Linux returns the following parameters:

dslu_name
Name of the downstream LU.

lu_sscp_sess_active
Specifies whether the LU-SSCP session is active. Possible values are:

YES The session is active.

NO The session is not active.

plu_sess_active
Specifies whether the PLU-SLU session is active. Possible values are:

YES The session is active.

NO The session is not active.

dspu_services
When the **query_downstream_lu** command is issued to a running node, this parameter specifies the services provided by the local node to the downstream LU.

Possible values are:

PU_CONCENTRATION
Downstream LU is served by SNA gateway.

DLUR Downstream LU is served by DLUR.

Session statistics are included for each of the three sessions (*lu_sscp_** for the LU-SSCP session, *ds_plu_** for the downstream PLU-SLU session, and *us_plu_** for the upstream PLU-SLU session). One of the session types precedes the following parameters:

rcv_ru_size
: Maximum RU size that can be received. This parameter is reserved in the LU-SSCP session statistics.

send_ru_size
: Maximum send RU size. This parameter is reserved in the LU-SSCP session statistics.

max_send_btu_size
: Maximum BTU size that can be sent.

max_rcv_btu_size
: Maximum BTU size that can be received.

max_send_pac_win
: Maximum size of the send pacing window on this session. This parameter is reserved in the LU-SSCP session statistics.

cur_send_pac_win
: Current size of the send pacing window on this session. This parameter is reserved in the LU-SSCP session statistics.

max_rcv_pac_win
: Maximum size of the receive pacing window on this session. This parameter is reserved in the LU-SSCP session statistics.

cur_rcv_pac_win
: Current size of the receive pacing window on this session. This parameter is reserved in the LU-SSCP session statistics.

send_data_frames
: Number of normal flow data frames sent.

send_fmd_data_frames
: Number of normal flow FMD data frames sent.

send_data_bytes
: Number of normal flow data bytes sent.

rcv_data_frames
: Number of normal flow data frames received.

rcv_fmd_data_frames
: Number of normal flow FMD data frames received.

rcv_data_bytes
: Number of normal flow data bytes received.

The following three parameters identify the Local Form Session Identifier (LFSID) for a session. The LFSID consists of the following parameters:

sidh
: Session ID high byte. (In the upstream PLU-SLU session statistics for an LU served by SNA gateway, this parameter is reserved.)

sidl
: Session ID low byte. (In the upstream PLU-SLU session statistics for an LU served by SNA gateway, this parameter is reserved.)

odai Origin Destination Assignor Indicator. (In the upstream PLU-SLU session statistics for an LU served by SNA gateway, this parameter is reserved.) Possible values are:

YES The BIND sender is the node containing the secondary link station.

NO The BIND sender is the node containing the primary link station.

ls_name
Link station name associated with statistics. (In the upstream PLU-SLU session statistics for an LU served by SNA gateway, this parameter is reserved.)

pacing_type
The type of receive pacing in use on this session. Possible values are:
```
NONE
FIXED
```

The following parameters are not preceded by a session type prefix:

description
A text string describing the downstream LU, as specified in the definition of the downstream LU.

This parameter is reserved for a DLUR-supported LU.

nau_address
Network accessible unit address of the downstream LU. This address is in the range 1–255.

dspu_name
Name of the PU associated with the downstream LU.

host_lu_name
For an LU supported by SNA gateway, the name of the host LU or host LU pool that the downstream LU uses.

If the downstream LU is used to communicate with a Communications Server for Linux Primary RUI application instead of a host, this field is set to the string `#PRIRUI#`.

This parameter is reserved for DLUR-served downstream LUs.

allow_timeout
Specifies whether this downstream LU allows its session with the upstream LU to timeout. Possible values are:

YES This downstream LU allows its session with the upstream LU to timeout.

NO This downstream LU does not allow its session with the upstream LU to timeout.

This field is ignored if the downstream LU is used to communicate with a Communications Server for Linux Primary RUI application instead of a host.

delayed_logon
Specifies whether this downstream LU uses delayed logon (the upstream LU is not activated until the user requests that it be activated). Possible values are:

YES This downstream LU uses delayed logon.

NO This downstream LU does not use delayed logon.

This field is ignored if the downstream LU is used to communicate with a Communications Server for Linux Primary RUI application instead of a host.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_LU_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the value specified in the *lu_name* parameter was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Function Not Supported

If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameter:

primary_rc

FUNCTION_NOT_SUPPORTED
The local node does not support SNA gateway or DLUR; support is defined by the *pu_conc_support* and *dlur_support* parameters in the node definition.

secondary_rc
(This parameter is not used.)

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_downstream_pu

The **query_downstream_pu** command returns information about downstream PUs that use SNA gateway, DLUR, or both. This command can be used to obtain information about a specific PU or about multiple PUs, depending on the options used.

This command must be issued to a running node.

Supplied Parameters

Parameter name	Type	Length	Default
[query_downstream_pu]			
num_entries	decimal		1
list_options	constant		LIST_INCLUSIVE
dspu_name	character	8	(null string)
dspu_services	constant		NONE

Supplied parameters are:

num_entries
: Maximum number of downstream PUs for which data should be returned. You can specify 1 to return data for a specific downstream PU, a number greater than 1 to return data for multiple downstream PUs, or `0` (zero) to return data for all downstream PUs.

list_options
: The position in the list of downstream PUs from which Communications Server for Linux begins to return data.

  Possible values are:

  FIRST_IN_LIST
  : Start at the first entry in the list

  LIST_INCLUSIVE
  : Start at the entry specified by the *dspu_name* parameter

  LIST_FROM_NEXT
  : Start at the entry immediately following the entry specified by the *dspu_name* parameter

dspu_name
: Name of the PU for which information is required, as specified on **define_*_ls**, or the name to be used as an index into the list of PUs. This value is ignored if *list_options* is set to `FIRST_IN_LIST`. The name is a type-A character string.

dspu_services
: DSPU services filter. Specifies whether to filter the returned information by the type of services provided to the PUs. Possible values are:

  PU_CONCENTRATION
  : Return information only about downstream PUs served by SNA gateway.

  DLUR Return information only about downstream PUs served by DLUR.

  NONE Return information about all downstream PUs.

Returned Parameters

Parameter name	Type	Length
dspu_name	character	8
description	character	31
ls_name	character	8
pu_sscp_sess_active	constant	
dspu_services	constant	
rcv_ru_size	decimal	
send_ru_size	decimal	
max_send_btu_size	decimal	
max_rcv_btu_size	decimal	
send_data_frames	decimal	
send_fmd_data_frames	decimal	

```
send_data_bytes          decimal
rcv_data_frames          decimal
rcv_fmd_data_frames      decimal
rcv_data_bytes           decimal
sidh                     hex number
sidl                     hex number
odai                     constant
pacing_type              constant
```

If the command executes successfully, Communications Server for Linux returns the following parameters:

dspu_name
Name of the downstream PU.

description
A text string describing the LS to the downstream PU, as specified in the definition of the LS.

ls_name
Name of the LS used to access the downstream PU.

pu_sscp_sess_active
Specifies whether the PU-SSCP session to the downstream PU is active. Possible values are:

YES The session is active.

NO The session is not active.

dspu_services
Specifies the type of services provided to the PU.

Possible values are:

PU_CONCENTRATION
Downstream PU is served by SNA gateway.

DLUR Downstream PU is served by DLUR.

rcv_ru_size
Maximum receive RU size; this parameter is reserved (set to `0`) if the downstream PU is served by SNA gateway.

send_ru_size
Maximum send RU size; this parameter is reserved (set to `0`) if the downstream PU is served by SNA gateway.

max_send_btu_size
Maximum BTU size that can be sent.

max_rcv_btu_size
Maximum BTU size that can be received.

send_data_frames
Number of normal flow data frames sent.

send_fmd_data_frames
Number of normal flow FMD data frames sent.

send_data_bytes
Number of normal flow data bytes sent.

rcv_data_frames
Number of normal flow data frames received.

rcv_fmd_data_frames
Number of normal flow FMD data frames received.

rcv_data_bytes
Number of normal flow data bytes received.

The following three parameters identify the Local Form Session Identifier (LSFID):

sidh Session ID high byte.

sidl Session ID low byte.

odai Origin Destination Assignor Indicator. Possible values are:

YES The BIND sender is the node containing the secondary link station.

NO The BIND sender is the node containing the primary link station.

pacing_type
The type of receive pacing in use on the PU-SSCP. This parameter will always be set to NONE.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_PU_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the value specified in the *dspu_name* parameter was not valid.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc

INVALID_PU_TYPE
The PU specified by the *dspu_name* parameter is not a downstream PU.

Function Not Supported

If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameters:

primary_rc

FUNCTION_NOT_SUPPORTED
The local node does not support SNA gateway or DLUR; support is defined by the *pu_conc_support* and *dlur_support* parameters in the node definition.

secondary_rc
(This parameter is not used.)

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_dspu_template

The **query_dspu_template** command returns information about defined downstream PU templates used for SNA gateway over implicit links.

Supplied Parameters

Parameter name	Type	Length	Default
[query_dspu_template]			
num_entries	decimal		1
list_options	constant		LIST_INCLUSIVE
template_name	character	8	(null string)

Supplied parameters are:

num_entries
Maximum number of entries for which data should be returned. You can specify 1 to return data for a specific template, a number greater than 1 to return data for multiple templates, or 0 (zero) to return data for all templates.

list_options
The position in the list of entries from which Communications Server for Linux begins to return data.

Possible values are:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the *template_name* parameter

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *template_name* parameter

template_name
Name of the DSPU template for which information is required, or the name to be used as an index into the list. Specify 1–8 locally displayable characters. This parameter is ignored if *list_options* is set to FIRST_IN_LIST.

Returned Parameters

Parameter name	Type	Length
template_name	character	8
description	character	
max_instance	decimal	
active_instance	decimal	
num_of_dslu_templates	decimal	

```
{dslu_template}
min_nau                 decimal
max_nau                 decimal
allow_timeout           constant
delayed_logon           constant
host_lu                 character      8
```

If the command executes successfully, the following parameters are returned:

template_name
Name of the DSPU template.

description
Resource description, as defined on the **define_dspu_template** command.

max_instance
The maximum number of instances of the template which can be active simultaneously.

active_instance
The number of instances of the template which are currently active.

num_of_dslu_templates
Number of downstream LU templates for this downstream PU template. Following this parameter are *num_of_dslu_templates* entries, one for each DSLU template.

Each `dslu_template` subrecord contains the following parameters:

min_nau
Minimum NAU address in the range of DSLU templates.

max_nau
Maximum NAU address in the range of DSLU templates.

allow_timeout
Indicates whether Communications Server for Linux is allowed to timeout host LUs used by this downstream LU if the session is left inactive for the timeout period specified on the host LU definition. Possible values are:

YES Communications Server for Linux is allowed to timeout host LUs used by this downstream LU.

NO Communications Server for Linux is not allowed to timeout host LUs used by this downstream LU.

This field is ignored if the downstream LUs are used to communicate with a Communications Server for Linux Primary RUI application instead of a host.

delayed_logon
Indicates whether Communications Server for Linux delays connecting the downstream LU to the host LU until the first data is received from the downstream LU. Instead, a simulated logon screen is sent to the downstream LU. Possible values are:

YES Communications Server for Linux delays connecting the downstream LU to the host LU.

NO Communications Server for Linux does not delay connecting the downstream LU to the host LU.

This field is ignored if the downstream LUs are used to communicate with a Communications Server for Linux Primary RUI application instead of a host.

host_lu_name
Name of the host LU or host LU pool onto which all the downstream LUs within the range will be mapped.

If the downstream LUs are used to communicate with a Communications Server for Linux Primary RUI application instead of a host, this field is set to the string `#PRIRUI#`.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
`PARAMETER_CHECK`

secondary_rc
Possible values are:

`INVALID_TEMPLATE_NAME`
The template specified in the *template_name* parameter was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_focal_point

The **query_focal_point** command returns information about the focal point for a specific Management Services category or about multiple focal points, depending on the options used.

Supplied Parameters

Parameter name	Type	Length	Default
[query_focal_point]			
num_entries	decimal		1
list_options	constant		LIST_INCLUSIVE
ms_category	character	8	(null string)

Supplied parameters are:

num_entries
Maximum number of focal point entries for which data should be

returned. You can specify 1 to return data for a specific focal point, a number greater than 1 to return data for multiple focal points, or 0 (zero) to return data for all focal points.

list_options
The position in the list of focal points from which Communications Server for Linux begins to return data.

Possible values are:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the *ms_category* parameter

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *ms_category* parameter

ms_category
Management Services category. This parameter is not used if *list_options* is set to FIRST_IN_LIST. This category may be one that is specified in *Systems Network Architecture: Management Services* or a user-defined category. A user-defined category name is a type-1134 character string.

Returned Parameters

Parameter name	Type	Length
ms_appl_name	character	8
ms_category	character	8
fp_fqcp_name	character	17
description	character	31
bkup_appl_name	character	8
bkup_fp_fqcp_name	character	17
implicit_appl_name	character	8
implicit_fp_fqcp_name	character	17
fp_type	constant	
fp_status	constant	
fp_routing	constant	
appl_name	character	8

If the command executes successfully, Communications Server for Linux returns the following parameters:

ms_appl_name
Name of the currently active focal point application. This name is one of the MS Discipline-Specific Application Program names specified in *Systems Network Architecture: Management Services*, or a user-defined category name.

ms_category
Management Services category. This category is one of the category names specified in *Systems Network Architecture: Management Services*, or a user-defined category name.

fp_fqcp_name
Fully qualified control point name of the focal point.

To revoke the existing focal point for the specified MS category, do not specify this parameter.

description
A text string describing the focal point, as specified in the definition of the focal point.

bkup_appl_name
Backup focal point application name. This name is one of the MS Discipline-Specific Application Programs specified in *Systems Network Architecture: Management Services*, or a user-defined category name.

bkup_fp_fqcp_name
Fully qualified control point name of the backup focal point.

implicit_appl_name
Name of the implicit focal point application, as specified using **define_focal_point**. This name is one of the MS Discipline-Specific Application Programs specified in *Systems Network Architecture: Management Services*, or a user-defined category name.

implicit_fp_fqcp_name
Fully qualified control point name of the implicit focal point, as specified using **define_focal_point**.

fp_type Type of focal point. For more information, refer to *Systems Network Architecture: Management Services*. Possible values are:

```
EXPLICIT_PRIMARY_FP
IMPLICIT_PRIMARY_FP
BACKUP_FP
DEFAULT_PRIMARY_FP
DOMAIN_FP
HOST_FP
NO_FP
```

fp_status
Status of the focal point. Possible values are:

ACTIVE The focal point is currently active.

NOT_ACTIVE
The focal point is currently not active.

PENDING
The focal point is pending active. This status occurs after an implicit request has been sent to the focal point and before the response has been received.

NEVER_ACTIVE
No focal point information is available for the specified category, although application registrations for the category have been accepted.

fp_routing
Specifies whether applications use default or direct routing to route traffic to the focal point. Possible values are:

DEFAULT
The MDS_MU is delivered to the focal point using default routing.

DIRECT The MDS_MU is routed on a session directly to the focal point.

appl_name
Name of the application registered for focal point category. This name is either one of the MS Discipline-Specific Application Programs specified in the *Systems Network Architecture: Management Services*, or a user-defined category name.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_MS_CATEGORY
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the value specified in the *ms_category* parameter was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Function Not Supported

If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameters:

primary_rc

FUNCTION_NOT_SUPPORTED
The local node does not support MS network management functions; support is defined by the *mds_supported* parameter on the **define_node** command.

secondary_rc
(This parameter is not used.)

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

query_global_log_type

The **query_global_log_type** command returns information about the types of events that Communications Server for Linux records in log files. It specifies default values that are used on all servers (unless they are overridden on a particular server by **set_log_type**); the **query_log_type** command can be used to determine the values being used on a particular server.

Communications Server for Linux always logs messages for problem events; you can specify whether to log messages for exception and audit events. For more information about logging messages, refer to the *Communications Server for Linux Diagnostics Guide*.

This command must be issued without specifying a node name.

Supplied Parameters

```
[query_global_log_type]
```

No parameters are supplied for this command.

Returned Parameters

```
Parameter name          Type
audit                   constant
exception               constant
succinct_audits         constant
succinct_errors         constant
```

If the command executes successfully, Communications Server for Linux returns the following parameters:

audit This parameter indicates whether audit messages are recorded. Possible values:

YES Audit messages are recorded.

NO Audit messages are not recorded.

exception
This parameter indicates whether exception messages are recorded. Possible values are:

YES Exception messages are recorded.

NO Exception messages are not recorded.

succinct_audits
This parameter indicates whether succinct logging or full logging is used in the audit log file. Possible values are:

YES Succinct logging is used in the audit log file. Each message in the log file contains a summary of the message header information (such as the message number, log type, and system name) and the message text string and parameters. To obtain more details about the cause of the log and any action required, you can use the **snahelp** utility.

NO Full logging is used in the audit log file. Each message in the log file includes a full listing of the message header information, the message text string and parameters, and additional information about the cause of the log and any action required.

succinct_errors
This parameter indicates whether succinct logging or full logging is used in the error log file; this applies to both exception logs and problem logs.

YES Succinct logging is used in the error log file. Each message in the log file contains a summary of the message header information (such as the message number, log type, and system name) and the message text string and parameters. To obtain more details about the cause of the log and any action required, you can use the **snahelp** utility.

NO Full logging is used in the error log file. Each message in the log file includes a full listing of the message header information, the message text string and parameters, and additional information about the cause of the log and any action required.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

NOT_CENTRAL_LOGGER
The command was issued to a specific node. It must be issued without specifying a node name.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_isr_session

The **query_isr_session** command returns information about the sessions for which a network node is providing intermediate session routing. The command can be used only if the Communications Server for Linux node is a network node; it is not valid if it is an end node or LEN node.

This command can be used to obtain information about a specific session or about a range of sessions, depending on the options used. When querying more than one session, the returned entries are ordered by *pcid* first and then by lexicographical ordering on *fqcp_name*.

This command must be issued to a running node.

Supplied Parameters

Parameter name	Type	Length	Default
[query_isr_session]			
num_entries	decimal		1
list_options	constant		SUMMARY + LIST_INCLUSIVE
session_type	constant		ISR_SESSIONS
pcid	hex array	8	(null array)
fqcp_name	character	17	(null string)

Supplied parameters are:

num_entries
Maximum number of sessions for which data should be returned. You can specify 1 to return data for a specific session, a number greater than 1 to return data for multiple sessions, or 0 (zero) to return data for all sessions.

list_options
The level of information required for each entry and the position in the list

of sessions from which Communications Server for Linux begins to return data. The list is ordered by *pcid* (numerically), and then by *fqcp_name*.

Specify the level of information required with one of the following values:

SUMMARY
Summary information only

DETAIL Detailed information

Use a + character to combine this value with one of the following values:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the *pcid* and *fqcp_name* parameters

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *pcid* and *fqcp_name* parameters

session_type
Specifies whether DLUR-maintained sessions or regular ISR sessions are being queried. Possible values are:

DLUR_SESSIONS
DLUR-maintained sessions are being queried.

ISR_SESSIONS
Regular ISR sessions are being queried.

pcid Procedure Correlator ID. This ID is an 8-byte hexadecimal string. This value is ignored if *list_options* is set to FIRST_IN_LIST.

fqcp_name
Fully qualified control point name of the session for which information is required, or the name to be used as an index into the list of sessions. This value is ignored if *list_options* is set to FIRST_IN_LIST. Specify 3–17 type-A characters consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character control point name.

Returned Parameters: Summary Information

Parameter name	Type	Length
pcid	hex array	8
fqcp_name	character	17

If the command executes successfully and you specified SUMMARY as the *list_options* parameter value, Communications Server for Linux returns the following parameters:

pcid Procedure Correlator ID. This ID is an 8-byte hexadecimal string.

fqcp_name
Fully qualified CP name.

Returned Parameters: Detailed Information

Parameter name	Type	Length
pcid	hex array	8
fqcp_name	character	17
trans_pri	constant	
cos_name	character	8
ltd_res	constant	

For each of the two sessions (primary and secondary), the following parameters are returned:

```
rcv_ru_size             decimal
send_ru_size            decimal
max_send_btu_size       decimal
max_rcv_btu_size        decimal
max_send_pac_win        decimal
cur_send_pac_win        decimal
send_rpc                decimal
max_rcv_pac_win         decimal
cur_rcv_pac_win         decimal
rcv_rpc                 decimal
send_data_frames        decimal
send_fmd_data_frames    decimal
send_data_bytes         decimal
send_fmd_data_bytes     decimal
rcv_data_frames         decimal
rcv_fmd_data_frames     decimal
rcv_data_bytes          decimal
rcv_fmd_data_bytes      decimal
sidh                    hex number
sidl                    hex number
odai                    constant
ls_name (or rtp_name)   character     8
rscv                    hex array     256
```

If the command executes successfully and you specified `DETAIL` as the *list_options* parameter value, Communications Server for Linux returns the following parameters:

pcid Procedure Correlator ID. This ID is an 8-byte hexadecimal string.

fqcp_name
Fully qualified CP name.

trans_pri **through** *ltd_res*
For more information about these parameters, see "query_session" on page 489.

rscv Route Selection control vector (RSCV) as defined in *Systems Network Architecture: Formats*. This control vector defines the session route through the network and is carried on the BIND. This RSCV is included only if the node's configuration indicates that RSCVs should be stored for ISR sessions.

For each of the two sessions (primary and secondary), the following parameters are returned:

rcv_ru_size
Maximum receive RU size.

send_ru_size
Maximum send RU size.

max_send_btu_size
Maximum BTU size that can be sent.

max_rcv_btu_size
Maximum BTU size that can be received.

max_send_pac_win
Maximum size of the send pacing window.

cur_send_pac_win
Current size of the send pacing window.

send_rpc
Send residual pacing count.

max_rcv_pac_win
Maximum size of the receive pacing window.

cur_rcv_pac_win
Current size of the receive pacing window.

rcv_rpc
Receive residual pacing count.

send_data_frames
Number of normal flow data frames sent.

send_fmd_data_frames
Number of normal flow FMD data frames sent.

send_data_bytes
Number of normal flow data bytes sent.

send_fmd_data_bytes
Number of normal flow FMD data bytes sent.

rcv_data_frames
Number of normal flow data frames received.

rcv_fmd_data_frames
Number of normal flow FMD data frames received.

rcv_data_bytes
Number of normal flow data bytes received.

rcv_fmd_data_bytes
Number of normal flow FMD data bytes received.

sidh Session ID high byte.

sidl Session ID low byte.

odai Origin Destination Assignor Indicator. When bringing up a session, the sender of the BIND sets this field to zero if the local node contains the primary link station, and sets it to one if the BIND sender is the node containing the secondary link station.

ls_name
Link station name or name of the RTP connection associated with statistics. This is an 8-byte string in a locally displayable character set. All 8 bytes are significant. This field can be used to correlate the intermediate session statistics with a particular link station.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
`PARAMETER_CHECK`

secondary_rc
Possible values are:

INVALID_FQPCID
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *pcid* parameter value was not valid.

State Check
No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Function Not Supported
If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameters:

primary_rc

INVALID_VERB
The local node is not a network node. This command can be used only at a network node.

secondary_rc
(This parameter is not used.)

Other Conditions
Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_kernel_memory_limit

The **query_kernel_memory_limit** command returns information about the amount of kernel memory that Communications Server for Linux is currently using, the maximum amount it has used, and the configured limit. This information enables you to check memory usage and set the limit appropriately to ensure that sufficient memory is available for Communications Server for Linux components and for other programs on the Linux computer.

You can specify the kernel memory limit when starting the Communications Server for Linux software (for more information, refer to the *Communications Server for Linux Administration Guide*), or modify it later when the node is running (using the **set_kernel_memory_limit** command).

Supplied Parameters

Parameter name	Type	Length	Default
[query_kernel_memory_limit]			
reset_max_used	constant		NO

Supplied parameter is:

reset_max_used
Specifies whether Communications Server for Linux resets the *max_used* value (after returning it on this command) to match the amount of memory currently allocated. This ensures that a subsequent **query_kernel_memory_limit** command will return the maximum amount used since this command was issued, rather than the maximum amount used since the system was started (or since the *max_used* value was last reset). Possible values are:

YES Reset the *max_used* value to match the current memory allocation.

NO Do not reset the *max_used* value.

Returned Parameters

```
Parameter name          Type
limit                   decimal
actual                  decimal
max_used                decimal
reset_max_used          constant
```

If the command executes successfully, Communications Server for Linux returns the following parameters:

limit The maximum amount of kernel memory, in bytes, that Communications Server for Linux is allowed to use at any one time. If a Communications Server for Linux component attempts to allocate kernel memory that would take the total amount of memory currently allocated above this limit, the allocation attempt will fail. A value of `0` (zero) indicates no limit.

actual The amount of kernel memory, in bytes, currently allocated to Communications Server for Linux components.

max_used
The maximum amount of kernel memory, in bytes, that has been allocated to Communications Server for Linux components at any time since the *max_used* parameter was last reset (as described for *reset_max_used*), or since the Communications Server for Linux software was started.

reset_max_used
Specifies whether Communications Server for Linux resets the *max_used* value (after returning it on this command) to match the amount of memory currently allocated. This ensures that a subsequent **query_kernel_memory_limit** command will return the maximum amount used since this command was issued, rather than the maximum amount used since the system was started (or since the *max_used* value was last reset). Possible values are:

YES Communications Server for Linux resets the *max_used* value to match the current memory allocation.

NO Communications Server for Linux does not reset the *max_used* value.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, “Common Return Codes from snaadmin Commands,” on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_local_lu

The **query_local_lu** command returns information about local LUs. This command can be used to obtain summary or detailed information about a specific LU or about multiple LUs, depending on the options used. It can also obtain information about the LU associated with the CP (the default LU).

Supplied Parameters

Parameter name	Type	Length	Default
[query_local_lu]			
num_entries	decimal		1
list_options	constant		SUMMARY + LIST_INCLUSIVE
lu_name	character	8	(null string)
lu_alias	character	8	(null string)
pu_name	character	8	(null string)

Supplied parameters are:

num_entries
Maximum number of LUs for which data should be returned. You can specify 1 to return data for a specific LU, a number greater than 1 to return data for multiple LUs, or 0 (zero) to return data for all LUs.

list_options
The level of information required for each entry and the position in the list of LUs from which Communications Server for Linux begins to return data. The list is in lexicographical order (regardless of the length of each name).

Specify the level of information required with one of the following values:

SUMMARY
Summary information only

DETAIL Detailed information

Use a + character to combine this value with one of the following values:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the *lu_name* or *lu_alias* parameter

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *lu_name* or *lu_alias* parameter

If FIRST_IN_LIST is specified, you can use a + character to include the following option:

LIST_BY_ALIAS
The list is returned in order of LU alias rather than LU name. This option is only valid if FIRST_IN_LIST is also specified. (For LIST_FROM_NEXT or LIST_INCLUSIVE, the list is in order of either LU alias or LU name, depending on which was specified as the index into the list.)

lu_name
Fully qualified name of the LU for which information is required, or the name to be used as an index into the list of LUs. The name is an 8-byte string. This value is ignored if *list_options* is set to `FIRST_IN_LIST`. To identify the LU by its alias instead of its name, do not specify this parameter; specify the alias in the *lu_alias* parameter. To identify the default LU, do not specify either parameter.

lu_alias
Alias of the LU for which information is required, or the name to be used as an index into the list of LUs. This value is ignored if *list_options* is set to `FIRST_IN_LIST`.

To identify the LU by its LU name instead of its alias, do not specify this parameter; specify the name in the *lu_name* parameter. To identify the default LU, do not specify either parameter.

pu_name
PU name filter. The name is a type-A character string starting with a letter. To return information only about LUs associated with a specific PU, specify the PU name. To return information without filtering on PU name, do not specify this parameter.

Returned Parameters: Summary Information

Parameter name	Type	Length
lu_name	character	8
lu_alias	character	8
description	character	31

If the command executes successfully and you specified `SUMMARY` as the *list_options* parameter value, the following parameters are returned:

lu_name
LU name.

lu_alias
LU alias.

description
A text string describing the local LU, as specified in the definition of the LU.

Returned Parameters: Detailed Information

Parameter name	Type	Length
lu_name	character	8
description	character	31
list_name	character	14
lu_alias	character	8
nau_address	decimal	
syncpt_support	constant	
lu_session_limit	decimal	
default_pool	constant	
pu_name	character	8
lu_attributes	constant	
allowed_sscp_id	decimal	
sys_name	character	128
timeout	decimal	

The following parameters are used only for dependent LUs. For independent LUs, they are reserved (set to binary zeros); you can obtain the equivalent information by issuing the **query_session** command for the appropriate session between this LU and the partner LU.

```
lu_sscp_sess_active      constant
appl_conn_active         constant
rcv_ru_size              decimal
send_ru_size             decimal
max_send_btu_size        decimal
max_rcv_btu_size         decimal
max_send_pac_win         decimal
cur_send_pac_win         decimal
max_rcv_pac_win          decimal
cur_rcv_pac_win          decimal
send_data_frames         decimal
send_fmd_data_frames     decimal
send_data_bytes          decimal
rcv_data_frames          decimal
rcv_fmd_data_frames      decimal
rcv_data_bytes           decimal
sidh                     hex number
sidl                     hex number
odai                     constant
pacing_type              constant
active_sscp_id           decimal
```

If the command executes successfully and you specified `DETAIL` as the *list_options* parameter value, the following parameters are returned:

lu_name
: LU name.

description
: A text string describing the local LU, as specified in the definition of the LU.

list_name
: Name of the security access list used by this local LU (defined using the **define_security_access_list** command). If this parameter is not set, the LU is available for use by any user.

lu_alias
: LU alias.

nau_address
: Network accessible unit (NAU) address of the LU. This address is in the range `1–255` if the LU is a dependent LU, or `0` (zero) if the LU is an independent LU.

syncpt_support **through** *timeout*
: For more information about these parameters, see "define_local_lu" on page 84. The parameter *allowed_sscp_id* returned on this command corresponds to the *sscp_id* parameter, if any, that was specified in the definition of the LU.

The following parameters are used only for dependent LUs. For independent LUs, they are reserved (set to binary zeros); you can obtain the equivalent information by issuing the **query_session** command for the appropriate session between this LU and the partner LU.

lu_sscp_sess_active
: Specifies whether the LU-SSCP session is active. Possible values are:

YES The session is active.

NO The session is not active.

appl_conn_active
Specifies whether an APPC or CPI-C transaction program (TP) is in a conversation with the host LU on a session from this local LU. Possible values are:

YES A conversation is in progress using the LU.

NO No conversation is in progress using the LU, although one or more TPs may have issued TP_STARTED for this LU.

rcv_ru_size
Maximum RU size that can be received.

send_ru_size
Maximum send RU size.

max_send_btu_size
Maximum BTU size that can be sent.

max_rcv_btu_size
Maximum BTU size that can be received.

max_send_pac_win
Maximum size of the send pacing window on this session.

cur_send_pac_win
Current size of the send pacing window on this session.

max_rcv_pac_win
Maximum size of the receive pacing window on this session.

cur_rcv_pac_win
Current size of the receive pacing window on this session.

send_data_frames
Number of normal flow data frames sent.

send_fmd_data_frames
Number of normal flow FMD data frames sent.

send_data_bytes
Number of normal flow data bytes sent.

rcv_data_frames
Number of normal flow data frames received.

rcv_fmd_data_frames
Number of normal flow FMD data frames received.

rcv_data_bytes
Number of normal flow data bytes received.

The following three parameters identify the Local Form Session Identifier (LFSID):

sidh Session ID high byte.

sidl Session ID low byte.

odai Origin Destination Assignor Indicator. Possible values are:

YES The BIND sender is the node containing the secondary link station.

NO The BIND sender is the node containing the primary link station.

pacing_type
The type of receive pacing in use on this session. Possible values are:
NONE
FIXED

active_sscp_id
Indicates the ID of the SSCP that has activated this LU. This is a 6-byte binary parameter.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_LU_ALIAS
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *lu_alias* parameter value was not valid.

INVALID_LU_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *lu_name* parameter value was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_local_topology

The Communications Server for Linux node maintains a local topology database that holds information about the TGs (transmission groups) to all adjacent nodes. The **query_local_topology** command returns information about these TGs. This command can be used to obtain summary or detailed information about a specific TG or about multiple TGs, depending on the options used.

This command must be issued to a running node.

Supplied Parameters

Parameter name	Type	Length	Default
[query_local_topology]			
num_entries	decimal		1
list_options	constant		SUMMARY + LIST_INCLUSIVE

```
dest                    character      17         (null string)
dest_type               constant                  LEARN_NODE
tg_num                  decimal                   0
```

Supplied parameters are:

num_entries
: Maximum number of TGs for which data should be returned. You can specify 1 to return data for a specific TG, a number greater than 1 to return data for multiple TGs, or 0 (zero) to return data for all TGs.

list_options
: The level of information required for each entry and the position in the list of TGs from which Communications Server for Linux begins to return data. The list is ordered by *dest*, then by *dest_type* (in the order of NETWORK_NODE, END_NODE, VRN), and finally in numerical order of *tg_num*.

  Specify the level of information required with one of the following values:

  SUMMARY
  : Summary information only

  DETAIL Detailed information

  Use a + character to combine this value with one of the following values:

  FIRST_IN_LIST
  : Start at the first entry in the list

  LIST_INCLUSIVE
  : Start at the entry specified by the combination of the *dest*, *dest_type*, and *tg_num* parameters

  LIST_FROM_NEXT
  : Start at the entry immediately following the entry specified by the combination of the *dest*, *dest_type*, and *tg_num* parameters

dest Fully qualified destination node name of the TG for which information is required, or the name to be used as an index into the list of TGs. This value is ignored if *list_options* is set to FIRST_IN_LIST. Specify 3–17 type-A characters consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character destination node name.

dest_type
: Node type of the destination node for this TG. This value is ignored if *list_options* is set to FIRST_IN_LIST. Possible values are:

  NETWORK_NODE
  : Network node (NN)

  END_NODE
  : End node (EN) or low-entry networking (LEN) node

  VRN Virtual routing node (VRN)

  LEARN_NODE
  : Unknown node type

tg_num
: Number associated with the TG. This value is ignored if *list_options* is set to FIRST_IN_LIST.

Returned Parameters: Summary Information

Parameter name	Type	Length
dest	character	17
dest_type	constant	
tg_num	decimal	

If the command executes successfully and you specified SUMMARY as the *list_options* parameter value, the following parameters are returned:

dest Fully qualified destination node name of the TG.

dest_type
Node type of the destination node for this TG. Possible values are:

NETWORK_NODE
Network node (NN)

VRN Virtual routing node (VRN)

END_NODE
End node (EN) or low-entry networking (LEN) node

tg_num
Number associated with the TG.

Returned Parameters: Detailed Information

Parameter name	Type	Length
dest	character	17
dest_type	constant	
tg_num	decimal	
dlc_data	hex array	32
rsn	decimal	
status	constant	
effect_cap	hex number	
connect_cost	decimal	
byte_cost	decimal	
security	constant	
prop_delay	constant	
user_def_parm_1	decimal	128
user_def_parm_2	decimal	128
user_def_parm_3	decimal	128
cp_cp_session_active	constant	
branch_link_type	constant	
branch_tg	constant	

If the command executes successfully and you specified DETAIL as the *list_options* parameter value, the following parameters are returned:

dest Fully qualified destination node name of the TG.

dest_type
Node type of the destination node for this TG. Possible values are:

NETWORK_NODE
Network node (NN)

VRN Virtual routing node (VRN)

END_NODE
End node (EN) or low-entry networking (LEN) node

tg_num
Number associated with the TG.

dlc_data
If *dest_type* is VRN, this parameter specifies the DLC address of the connection to the VRN. The number of bytes in the address depends on the DLC type. This parameter is not used otherwise.

For Token Ring or Ethernet, the address is in two parts: a 6-byte MAC address and a 1-byte local SAP address. The bit ordering of the MAC address may not be in the expected format. For information about converting between the two address formats, see *Bit ordering in MAC addresses* in "define_tr_ls, define_ethernet_ls" on page 209.

rsn Resource sequence number assigned by the owning network node.

status Specifies the status of the TG. Possible values (which can be combined with a + character) are:

TG_OPERATIVE
Transmission group link is operative.

TG_CP_CP_SESSIONS
Transmission group link between CP-CP sessions.

TG_QUIESCING
Transmission group link is quiescing.

TG_HPR Transmission group supports High Performance Routing (HPR) protocols.

TG_RTP Transmission group supports Rapid Transport Protocol (RTP).

effect_cap **through** *user_def_parm_3*
TG characteristics. For more information about these parameters, see "define_tr_ls, define_ethernet_ls" on page 209.

cp_cp_session_active
Specifies whether the owning node's contention winner CP-CP session is active. Possible values are:

YES The CP-CP session is active.

NO The CP-CP session is not active.

UNKNOWN
The CP-CP session status is unknown.

branch_link_type
This parameter applies only if the node is a Branch Network Node; it is reserved otherwise.

Specifies the branch link type of this TG. Possible values are:

UPLINK The TG is an uplink.

DOWNLINK
The TG is a downlink to an End Node.

DOWNLINK_TO_BRNN
The TG is a downlink to a Branch Network Node that appears as an End Node from the perspective of the local node.

OTHER The TG type is a link to a VRN.

NOT_SUPPORTED
This parameter does not apply because the local node is not a Branch Network Node.

branch_tg
This parameter applies only if the node is a Network Node; it is reserved otherwise.

Specifies whether the TG is a branch TG. Possible values are:

YES The TG is a branch TG.

NO The TG is not a branch TG.

UNKNOWN
The TG type is unknown.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_TG
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *tg_num* parameter value was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_log_file

The **query_log_file** command enables you to determine the name of the file that Communications Server for Linux uses to record audit, error, or usage log messages, the name of the backup log file, and the file size at which log information is copied to the backup file.

Supplied Parameters

Parameter name	Type	Length	Default
[query_log_file]			
log_file_type	constant		ERROR

Supplied parameters are:

log_file_type
The type of log file to be queried. Possible values are:

AUDIT Audit log file (audit messages only)

ERROR Error log file (problem and exception messages)

USAGE Usage log file (current and peak usage of Communications Server for Linux resources).

Returned Parameters

Parameter name	Type	Length
file_name	character	80
backup_file_name	character	80
file_size	decimal	

If the command executes successfully, Communications Server for Linux returns the following parameters:

file_name
Name of the log file.

If no path is included, the file is stored in the default directory for diagnostics files, **/var/opt/ibm/sna**. If a path is included, this path is either a full path (starting with a **/** character) or the path relative to the default directory.

backup_file_name
Name of the backup log file. When the log file reaches the size specified by *file_size*, Communications Server for Linux copies the current contents of the log file to this file and then clears the log file. You can also request a backup at any time using **set_log_file**.

If no path is included, the backup log file is stored in the default directory for diagnostics files, **/var/opt/ibm/sna**. If a path is included, it is either a full path (starting with a **/** character) or the path relative to the default directory.

file_size
The maximum size of the log file specified by *log_file_type*. When a message written to the file causes the file size to exceed this limit, Communications Server for Linux clears the backup log file, copies the current contents of the log file to the backup log file, and then clears the log file. The maximum amount of disk space taken up by log files is approximately twice the value of *file_size*.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_log_type

The **query_log_type** command returns information about the categories of log messages that Communications Server for Linux records in log files, and whether these categories of log messages are the default settings specified on **set_global_log_type** or local settings specified by a previous **set_log_type** command.

Communications Server for Linux always logs messages for problem events; you can specify whether to log messages for exception and audit events. For more information about logging messages, refer to the *Communications Server for Linux Diagnostics Guide*.

Supplied Parameters

```
[query_log_type]
```

No parameters are supplied for this command.

Returned Parameters

Parameter name	Type
override	constant
audit	constant
exception	constant
succinct_audits	constant
succinct_errors	constant

If the command executes successfully, Communications Server for Linux returns the following parameters:

override
Specifies whether the log types and succinct or full logging options returned on this command are the global log types specified on **set_global_log_type**, or the local values specified on **set_log_type**. Possible values are:

YES The *audit*, *exception*, and *succinct_** parameters returned are local settings overriding the global settings.

NO The *audit*, *exception*, and *succinct_** parameters returned are the global settings, which are not being overridden.

audit This parameter indicates whether audit messages are recorded. Possible values are:

YES Audit messages are recorded.

NO Audit messages are not recorded.

exception
This parameter indicates whether exception messages are recorded. Possible values are:

YES Exception messages are recorded.

NO Exception messages are not recorded.

succinct_audits
This parameter indicates whether succinct logging or full logging is used in the audit log file. Possible values are:

YES Succinct logging is used in the audit log file. Each message in the

log file contains a summary of the message header information (such as the message number, log type, and system name), and the message text string and parameters. To obtain more details about the cause of the log and any action required, you can use the **snahelp** utility.

NO Full logging is used in the audit log file. Each message in the log file includes a detailed listing of the message header information, the message text string and parameters, and additional information about the cause of the log and any action required.

succinct_errors
This parameter indicates whether succinct logging or full logging is used in the error log file; this applies to both exception logs and problem logs. Possible values are:

YES Succinct logging is used in the error log file. Each message in the log file contains a summary of the message header information (such as the message number, log type, and system name), and the message text string and parameters. To obtain more details about the cause of the log and any action required, you can use the **snahelp** utility.

NO Full logging is used in the error log file. Each message in the log file includes a detailed listing of the message header information, the message text string and parameters, and additional information about the cause of the log and any action required.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

query_ls

The **query_ls** command returns a list of information about the link stations defined at the node. This information is structured as determined data (data dynamically gathered during execution, and returned only if the LS is active) and defined data (data supplied in the definition of the LS).

This command can be used to obtain summary or detailed information about a specific link station or about multiple link stations, depending on the options used. For multiple link stations, the information is returned in a separate entry for each link station.

Supplied Parameters

Parameter name	Type	Length	Default
[query_ls]			
num_entries	decimal		1
list_options	constant		SUMMARY + LIST_INCLUSIVE
ls_name	character	8	(null string)
port_name	character	8	(null string)

Supplied parameters are:

num_entries
: Maximum number of link stations for which data should be returned. You can specify 1 to return data for a specific link station, a number greater than 1 to return data for multiple link stations, or 0 (zero) to return data for all link stations.

list_options
: The level of information required for each entry and the position in the list of link stations from which Communications Server for Linux begins to return data.

  Specify the level of information required with one of the following values:

  SUMMARY
  : Summary information only

  DETAIL Detailed information

  Use a + character to combine this value with one of the following values:

  FIRST_IN_LIST
  : Start at the first entry in the list

  LIST_INCLUSIVE
  : Start at the entry specified by the *ls_name* parameter

  LIST_FROM_NEXT
  : Start at the entry immediately following the entry specified by the *ls_name* parameter

ls_name
: Link station name. This value is ignored if *list_options* is set to FIRST_IN_LIST.

port_name
: Port name filter. To return information only on link stations associated with a specific port, specify the name of the port. To return information about all link stations without filtering on the port name, do not specify this parameter.

Returned Parameters: Summary Information

Parameter name	Type	Length
ls_name	character	8
description	character	31
dlc_type	constant	
state	constant	
act_sess_count	decimal	
det_adj_cp_name	character	17
det_adj_cp_type	constant	
port_name	character	8
adj_cp_name	character	17
adj_cp_type	constant	

If the command executes successfully and you specified SUMMARY as the *list_options* parameter value, Communications Server for Linux returns the following parameters:

ls_name
Link station name.

description
A text string describing the LS, as specified in the definition of the LS.

dlc_type
Type of the DLC. Possible values are:

SDLC Synchronous data link control

X25 X.25 QLLC (qualified link level control)

TR Token Ring

ETHERNET
Ethernet

MPC Multipath Channel (MPC), Communications Server for Linux on System z only

HPRIP Enterprise Extender (HPR/IP)

state State of this link station. Possible values are:

ACTIVE The LS is active.

NOT_ACTIVE
The LS is not active.

PENDING_ACTIVE
The LS is being activated.

PENDING_INACTIVE
The LS is being deactivated.

PENDING_ACTIVE_BY_LR
The LS has failed (or an attempt to activate it has failed) and Communications Server for Linux is attempting to reactivate it.

act_sess_count
The total number of active sessions (both endpoint and intermediate) using the link.

det_adj_cp_name
Fully qualified name of the adjacent control point. This name is usually determined during activation; it is null if the LS is inactive. However, for an LS to a back-level LEN node (specified by the *adj_cp_type* parameter on **define_*_ls**), this name is taken from the LS definition and is not determined during activation.

det_adj_cp_type
Type of the adjacent node. The node type is usually determined during activation; it is null if the LS is inactive. However, for an LS to a back-level LEN node (specified by the *adj_cp_type* parameter on **define_*_ls**), the node type is taken from the LS definition and is not determined during activation.

Possible values are:

LEARN_NODE
Node type is unknown or LS is inactive.

END_NODE
An End Node, a Branch Network Node acting as an End Node from the local node's perspective, or a LEN node that includes the Network Name CV in its XID3.

NETWORK_NODE
A Network Node, or a Branch Network Node acting as a Network Node from the local node's perspective.

VRN Virtual routing node.

port_name
Name of the port associated with this link station.

adj_cp_name
Fully qualified name of the adjacent control point; this parameter is null for an implicit link.

adj_cp_type
Type of the adjacent control point. Possible values are:

LEARN_NODE
Node type is unknown or LS is inactive.

END_NODE
An End Node, a Branch Network Node acting as an End Node from the local node's perspective, or a LEN node that includes the Network Name CV in its XID3.

NETWORK_NODE
A Network Node, or a Branch Network Node acting as a Network Node from the local node's perspective.

BACK_LEVEL_LEN_NODE
Back-level LEN node (one that does not include the Network Name CV in its XID3).

HOST_XID3
Host node; Communications Server for Linux responds to a polling XID from the node with a format 3 XID.

HOST_XID0
Host node; Communications Server for Linux responds to a polling XID from the node with a format 0 XID.

DSPU_XID
Downstream PU; Communications Server for Linux includes XID exchange in link activation. The *dspu_name* and *dspu_services* parameters are also returned.

DSPU_NOXID
Downstream PU; Communications Server for Linux does not include XID exchange in link activation. The *dspu_name* and *dspu_services* parameters are also returned.

VRN Virtual routing node.

Returned Parameters: Detailed Information

Parameter name	Type	Length
ls_name	character	8
dlc_type	constant	
state	constant	
sub_state	constant	

```
act_sess_count                decimal
det_adj_cp_name               character      17
det_adj_cp_type               constant
dlc_name                      character      8
dynamic                       constant
migration                     constant
tg_num                        decimal
in_xid_bytes                  decimal
in_msg_bytes                  decimal
in_xid_frames                 decimal
in_msg_frames                 decimal
out_xid_bytes                 decimal
out_msg_bytes                 decimal
out_xid_frames                decimal
out_msg_frames                decimal
in_invalid_sna_frames         decimal
in_session_control_frames     decimal
out_session_control_frames    decimal
good_xids                     decimal
bad_xids                      decimal
start_time                    decimal
stop_time                     decimal
up_time                       decimal
current_state_time            decimal
deact_cause                   constant
determined_hpr_support        constant
anr_label                     hex array      2
determined_hpr_link_lvl_error  contant
auto_act                      constant
ls_type                       constant
det_branch_link_type          constant
adj_cp_is_brnn                constant
node_id                       hex array      4
active_isr_count              decimal
active_lu_sess_count          decimal
active_sscp_sess_count        decimal
reverse_anr_label             hex array      8
local_address                 hex array      32
actual_max_send_btu_size      decimal
description                   character      31
port_name                     character      8
adj_cp_name                   character      17
adj_cp_type                   constant
auto_act_supp                 constant
tg_number                     decimal
limited_resource              constant
solicit_sscp_sessions         constant
pu_name                       character      8
disable_remote_act            constant
default_nn_server             constant
hpr_supported                 constant
hpr_link_lvl_error            constant
link_deact_timer              decimal
use_default_tg_chars          constant
ls_attributes                 constant
adj_node_id                   hex array      4
local_node_id                 hex array      4
cp_cp_sess_support            constant
effect_cap                    decimal
connect_cost                  decimal
byte_cost                     decimal
security                      constant
prop_delay                    constant
user_def_parm_1               decimal
user_def_parm_2               decimal
user_def_parm_3               decimal
target_pacing_count           decimal
```

```
max_send_btu_size             decimal
ls_role                       constant
max_ifrm_rcvd                 decimal
dlus_retry_timeout            decimal
dlus_retry_limit              decimal
conventional_lu_compression   constant
branch_link_type              constant
adj_brnn_cp_support           constant
dddlu_offline_supported       constant
initially_active              constant
dddlu_offline_supported       constant
react_timer                   decimal
react_timer_retry             decimal
```

For SDLC, the following parameters are included. For more information about these parameters, see "define_sdlc_ls" on page 162.

```
address                       hex number
poll_frame                    constant
```

For QLLC, the following parameters are included. For more information about these parameters, see "define_qllc_ls" on page 135.

```
address                       character      14
vc_type                       constant
fac                           character      32
pvc_id                        decimal
sn_id                         character      4
cud                           character      16
```

For Token Ring or Ethernet, the following parameters are included. For more information about these parameters, see "define_tr_ls, define_ethernet_ls" on page 209.

```
mac_address                   hex array      6
lsap_address                  hex number
```

For Token Ring / Ethernet :

```
xid_timer                     decimal
xid_timer_retry               decimal
test_timeout                  decimal
test_timer_retry              decimal
ack_timeout                   decimal
p_bit_timeout                 decimal
t2_timeout                    decimal
rej_timeout                   decimal
busy_state_timeout            decimal
idle_timeout                  decimal
max_retry                     decimal
```

For Enterprise Extender (HPR/IP), the following parameters are included. The parameter *determined_ip_address* is described below; for more information about the remaining parameters, see "define_ip_ls" on page 67.

```
determined_ip_address         character
lsap_address                  hex number
remote_ip_host                character  100
ack_timeout                   decimal
max_retry                     decimal
liveness_timeout              decimal
short_hold_mode               constant
```

If the command executes successfully and you specified `DETAIL` as the *list_options* parameter value, Communications Server for Linux returns the following parameters:

ls_name
Link station name.

dlc_type
Type of the DLC. Possible values are:

SDLC Synchronous data link control

X25 X.25 QLLC (qualified link level control)

TR Token Ring

ETHERNET
Ethernet

MPC Multipath Channel (MPC), Communications Server for Linux on System z only

HPRIP Enterprise Extender (HPR/IP)

state State of this link station. Possible values are:

ACTIVE The LS is active.

NOT_ACTIVE
The LS is not active.

PENDING_ACTIVE
The LS is being activated.

PENDING_INACTIVE
The LS is being deactivated.

PENDING_ACTIVE_BY_LR
The LS has failed (or an attempt to activate it has failed) and Communications Server for Linux is attempting to reactivate it.

sub_state
This parameter provides more detailed information about the state of this link station. Possible values are:

SENT_CONNECT_OUT
The local node has requested that initial contact be established.

PENDING_XID_EXCHANGE
Initial contact has been established (for example, TEST exchange on a LAN device) and the XID negotiation is in progress.

SENT_ACTIVATE_AS
Creating internal processes to handle the link.

SENT_SET_MODE
Waiting for a response to SNRM/SABME from the remote node.

ACTIVE The link is fully active.

SENT_DEACTIVATE_AS_ORDERLY
Destroying internal processes.

SENT_DISCONNECT
The local node has sent a DISC frame to the remote node.

WAITING_STATS
The link has been disconnected. Final link statistics have been requested but not yet received.

RESET The link is inactive.

act_sess_count
The total number of active sessions (both endpoint and intermediate) using the link.

det_adj_cp_name
Fully qualified name of the adjacent control point. This name is usually determined during activation; it is null if the LS is inactive. However, for an LS to a back-level LEN node (specified by the *adj_cp_type* parameter on **define_*_ls**), this name is taken from the LS definition and is not determined during activation.

det_adj_cp_type
Type of the adjacent node, determined during link activation. Possible values are as follows:

LEARN_NODE
Node type is unknown or LS is inactive.

END_NODE
An End Node, a Branch Network Node acting as an End Node from the local node's perspective, or a LEN node that includes the Network Name CV in its XID3.

NETWORK_NODE
A Network Node, or a Branch Network Node acting as a Network Node from the local node's perspective.

VRN Virtual routing node.

The node type is usually determined during activation; it is null if the LS is inactive. However, for an LS to a back-level LEN node (specified by the *adj_cp_type* parameter on **define_*_ls**), the node type is taken from the LS definition and is not determined during activation.

dlc_name
Name of the DLC used by the LS.

dynamic
Specifies whether the link was dynamically defined. Possible values are:

YES The link was dynamically defined (in response to a connection request from the adjacent node or to dynamically connect to another node across a connection network).

NO The link was explicitly defined as part of the Communications Server for Linux configuration.

migration
Specifies whether the adjacent node is a migration level node (such as a LEN node) or a full APPN network node or end node. Possible values are:

YES The adjacent node is a migration-level node.

NO The adjacent node is a network node or end node.

UNKNOWN
The adjacent node level is unknown.

tg_num
Number associated with the TG.

in_xid_bytes
Total number of XID bytes received on this link station.

in_msg_bytes
Total number of data bytes received on this link station.

in_xid_frames
Total number of XID frames received on this link station.

in_msg_frames
Total number of data frames received on this link station.

out_xid_bytes
Total number of XID bytes sent on this link station.

out_msg_bytes
Total number of data bytes sent on this link station.

out_xid_frames
Total number of XID frames sent on this link station.

out_msg_frames
Total number of data frames sent on this link station.

in_invalid_sna_frames
Total number of SNA frames received on this link station that were not valid.

in_session_control_frames
Total number of session control frames received on this link station.

out_session_control_frames
Total number of session control frames sent on this link station.

good_xids
Total number of successful XID exchanges that have occurred on this link station since it was started.

bad_xids
Total number of unsuccessful XID exchanges that have occurred on this link station since it was started.

start_time
The time, in hundredths of a second, since system startup, that the link station was last activated (when the mode setting commands completed).

stop_time
The time, in hundredths of a second, since system startup, that the link station was last deactivated.

up_time
Total time, in hundredths of a second that this link station has been active since system startup.

current_state_time
Total time, in hundredths of a second that this link station has been in its current state.

deact_cause
The cause of the last deactivation of the link station. Possible values are:

NONE The link station has never been deactivated.

DEACT_OPER_ORDERLY
The link station was deactivated as a result of an orderly stop (on the **stop_ls** command) from an operator.

DEACT_OPER_IMMEDIATE
The link station was deactivated as a result of an immediate stop (on the **stop_ls** command) from an operator.

DEACT_AUTOMATIC
The link station was automatically deactivated because there were no more sessions using the link station.

DEACT_FAILURE
The link station was deactivated because of a failure.

determined_hpr_support
Level of High Performance Routing (HPR) supported on this transmission group (TG), taking account of the capabilities of the local and adjacent nodes. Possible values are:

NONE This TG does not support HPR protocols.

BASE This TG supports base level HPR.

RTP This TG supports Rapid Transport Protocols (RTP).

anr_label
The HPR automatic network routing (ANR) label allocated to the local link.

determined_hpr_link_lvl_error
Specifies whether link-level error recovery is being used for HPR traffic on the link.

auto_act
Specifies whether the link currently allows remote activation or activation on demand. This parameter is set to NONE if neither is allowed, or to one or both of the following values (combined with a + character):

AUTO_ACT
The link can be activated on demand by the local node when a session requires it.

REMOTE_ACT
The link can be activated by the remote node.

ls_type Specifies how this link was defined or discovered. Possible values are:

LS_DEFINED
The link station was defined explicitly by a Communications Server for Linux administration program.

LS_DYNAMIC
The link station was created when the local node connected to another node through a connection network.

LS_TEMPORARY
The link station was created temporarily to process an incoming call, but has not yet become active.

LS_IMPLICIT
The link station was defined implicitly when Communications Server for Linux received an incoming call that it could not match to a defined link station.

LS_DLUS_DEFINED
The link station is a dynamic link station to a DLUR-served downstream PU, and was defined when the local node received an ACTPU from a DLUS.

det_branch_link_type
This parameter applies only if the local node is a Branch Network Node; it is not used otherwise.

Specifies the branch link type of this link. Possible values are:

UPLINK The link is an uplink.

DOWNLINK
The link is a downlink.

OTHERLINK
The link is to a VRN.

UNKNOWN_LINK_TYPE
The branch link type is unknown.

BRNN_NOT_SUPPORTED
The link supports PU 2.0 traffic only.

adj_cp_is_brnn
Specifies whether the adjacent node is a Branch Network Node. Possible values are:

YES The adjacent node is a Branch Network Node.

NO The adjacent node is not a Branch Network Node.

UNKNOWN
The adjacent node type is unknown.

node_id
Node ID received from the adjacent node during XID exchange.

active_isr_count
Number of active intermediate sessions using this link.

active_lu_sess_count
Number of active LU-LU sessions using this link.

active_sscp_sess_count
Number of active PU-SSCP sessions using this link.

reverse_anr_label
The Reverse Automatic Network Routing (ANR) label for this link station.

local_address
The local address of this link station. For an Enterprise Extender (HPR/IP) link, this is shown as as a dotted-decimal IP address (such as `193.1.11.100`.

actual_max_send_btu_size
Negotiated maximum send BTU size.

description
A text string describing the LS, as specified in the definition of the LS.

port_name
Name of the port associated with this link station. If the link is to a virtual routing node (VRN), this parameter specifies the name of the actual port used to connect to the VRN (as specified in the **define_cn** command).

adj_cp_name
Fully qualified name of the adjacent control point. This parameter is used only if *adj_cp_type* specifies that the adjacent node is an APPN node or a back-level LEN node.

adj_cp_type
Adjacent node type. Possible values are:

LEARN_NODE
APPN-capable node; the node type will be identified during XID exchange.

END_NODE
An End Node, a Branch Network Node acting as an End Node from the local node's perspective, or a LEN node that includes the Network Name CV in its XID3.

NETWORK_NODE
A Network Node, or a Branch Network Node acting as a Network Node from the local node's perspective.

BACK_LEVEL_LEN_NODE
Back-level LEN node (one that does not include the Network Name CV in its XID3).

HOST_XID3
Host node. Communications Server for Linux responds to a polling XID from the node with a format 3 XID.

HOST_XID0
Host node. Communications Server for Linux responds to a polling XID from the node with a format 0 XID.

DSPU_XID
Downstream PU. Communications Server for Linux includes XID exchange in link activation.

DSPU_NOXID
Downstream PU. Communications Server for Linux does not include XID exchange in link activation.

auto_act_supp
Specifies whether the link can be automatically activated when required by a session. Possible values are:

YES The link can be automatically activated.

NO The link cannot be automatically activated.

tg_number
Preassigned TG number, used to represent the link when the link is activated. This parameter is used only if the adjacent node is an APPN node (*adj_cp_type* is either NETWORK_NODE or END_NODE); it is ignored otherwise. The value 0 (zero) indicates that the TG number is not preassigned and is negotiated when the link is activated.

limited_resource
Specifies whether this link station is a limited resource and is automatically deactivated when there are no sessions using the link. Link stations on a nonswitched port cannot be configured as limited resource. Possible values are:

NO The link is not a limited resource and is not automatically deactivated.

NO_SESSIONS
The link is a limited resource and is automatically deactivated when no active sessions are using it.

INACTIVITY
The link is a limited resource and is automatically deactivated when no active sessions are using it, or when no data has flowed on the link for the time period specified by the *link_deact_timer* parameter.

solicit_sscp_sessions
Specifies whether to request the adjacent node to initiate sessions between the SSCP and the local CP and dependent LUs.

This parameter is used only if the adjacent node is an APPN node (the *adj_cp_type* parameter is either NETWORK_NODE or END_NODE); it is ignored otherwise. If the adjacent node is a host (the *adj_cp_type* parameter is either HOST_XID3 or HOST_XID0), Communications Server for Linux always requests the host to initiate SSCP sessions.

Possible values are:

YES Request the adjacent node to initiate SSCP sessions.

NO Do not request the adjacent node to initiate SSCP sessions.

pu_name
Name of the local PU that uses this link. This parameter is used only if *adj_cp_type* is set to HOST_XID3 or HOST_XID0, or if *solicit_sscp_sessions* is set to YES.

disable_remote_act
Specifies whether the LS can be activated by a remote node. Possible values are:

YES The LS can be activated only by the local node; if the remote node attempts to activate it, Communications Server for Linux will reject the attempt.

NO The LS can be activated by the remote node.

default_nn_server
For an end node, this parameter specifies whether this is a link supporting CP-CP sessions to a network node that can act as the local node's network node server. When the local node has no CP-CP sessions to a network node server and needs to establish them, the local node checks this parameter on its defined link stations to find a suitable LS to activate. This enables you to specify which adjacent NNs are suitable to act as the NN server (for example, to avoid using NNs that are accessed by expensive or slow links).

Possible values are:

YES This link supports CP-CP sessions to a network node that can act as the local node's NN server; the local node can automatically activate this link if it needs to contact an NN server.

NO This link does not support CP-CP sessions to a network node that can act as the local node's NN server; the local node cannot automatically activate this link if it needs to contact an NN server.

If the local node is not an end node, this parameter is not used.

hpr_supported
Specifies whether HPR is supported on this link. Possible values are:

YES HPR is supported on this link.

NO HPR is not supported on this link.

hpr_link_lvl_error
Specifies whether HPR traffic should be sent on this link using link-level error recovery. This parameter should be ignored unless *hpr_supported* is set to YES. Possible values are:

YES HPR traffic should be sent on this link using link-level error recovery.

NO HPR traffic should not be sent on this link using link-level error recovery.

link_deact_timer
Limited resource link deactivation timer, in seconds. A limited resource link is automatically deactivated if not data flows over the link for the time specified by this parameter. This parameter is not used if *limited_resource* is set to any value other than INACTIVITY.

use_default_tg_chars
Specifies whether the default TG characteristics supplied in the port definition are used. Possible values are:

YES Use the default TG characteristics; ignore *effect_cap* through *user_def_parm_3* parameters on this command.

NO Use *effect_cap* through *user_def_parm_3* parameters returned on this command.

ls_attributes
Attributes of the remote system with which Communications Server for Linux is communicating.

This parameter is usually set to SNA, unless you are communicating with a host of one of the other types listed below. Possible values are:

SNA Standard SNA host.

FNA Fujitsu Network Architecture (VTAM-F) host.

HNA Hitachi Network Architecture host.

SUPPRESS_CP_NAME
Suppress the CP name associated with the remote node.

If this LS is to a back-level LEN node that cannot accept the Network Name CV in the format 3 XID it receives, a + character is used to combine the value SNA, FNA, or HNA with SUPPRESS_CP_NAME (for example, SNA+SUPPRESS_CP_NAME). If the LS is to any other node type, or to a back-level node that can accept the Network Name CV, the option SUPPRESS_CP_NAME is not used.

adj_node_id
Node ID of adjacent node. This ID is a 4-byte hexadecimal string; a value of 4 zeros indicates that node ID checking is disabled.

local_node_id
Node ID sent in XIDs on this LS. This ID is a 4-byte hexadecimal string; a value of 4 zeros indicates that Communications Server for Linux uses the node ID specified in **define_node**.

cp_cp_sess_support
Specifies whether CP-CP sessions are supported. Possible values are:

YES CP-CP sessions are supported.

NO CP-CP sessions are not supported.

effect_cap
A decimal value representing the line speed in bits per second.

connect_cost
Cost per connect time.

byte_cost
Cost per byte.

security
Security level of the network. Possible values are:

SEC_NONSECURE
No security.

SEC_PUBLIC_SWITCHED_NETWORK
Data is transmitted over a public switched network.

SEC_UNDERGROUND_CABLE
Data is transmitted over secure underground cable.

SEC_SECURE_CONDUIT
Data is transmitted over a line in a secure conduit that is not guarded.

SEC_GUARDED_CONDUIT
Data is transmitted over a line in a conduit that is protected against physical tapping.

SEC_ENCRYPTED
Data is encrypted before transmission over the line.

SEC_GUARDED_RADIATION
Data is transmitted over a line that is protected against physical and radiation tapping.

SEC_MAXIMUM
Maximum security.

prop_delay
Propagation delay. The time that a signal takes to travel the length of the link. Possible values are:

PROP_DELAY_MINIMUM
Minimum propagation delay.

PROP_DELAY_LAN
Delay is less than .5 microseconds (typical for a LAN).

PROP_DELAY_TELEPHONE
Delay is in the range .5–50 microseconds (typical for a telephone network).

PROP_DELAY_PKT_SWITCHED_NET
Delay is in the range 50–250 microseconds (typical for a packet-switched network).

PROP_DELAY_SATELLITE
Delay is greater than 250 microseconds (typical for a satellite link).

PROP_DELAY_MAXIMUM
Maximum propagation delay.

user_def_parm_1 **through** *user_def_parm_3*
User-defined parameters.

target_pacing_count
Indicates the desired pacing window size.

max_send_btu_size
Maximum BTU size that can be sent. This value includes the length of the TH and RH (total 9 bytes), as well as the RU.

ls_role The determined LS role of this link. This role is usually taken from the definition of the port that owns the LS (or from the definition of the LS, if this overrides the LS role in the port definition). However, if the LS role is defined as negotiable, it will be negotiated to either primary or secondary while the LS is active, so (for an active LS) this parameter returns the negotiated role currently in use and not the defined role. Possible values are:

LS_PRI Primary

LS_SEC Secondary

LS_NEG Negotiable

max_ifrm_rcvd
Maximum number of I-frames that can be received by the local link stations before an acknowledgment is sent.

dlus_retry_timeout
Time interval in seconds between attempts to contact the DLUS and backup DLUS.

dlus_retry_limit
Number of attempts to recontact a DLUS after an initial failure.

conventional_lu_compression
Specifies whether data compression is requested for LU 0–3 sessions on this link. This parameter is used only if this link carries LU 0–3 traffic; it does not apply to LU 6.2 sessions. Possible values are:

YES Data compression should be used for LU 0–3 sessions on this link if the host requests it.

NO Data compression should not be used for LU 0–3 sessions on this link.

branch_link_type
This parameter applies only if the local node is a Branch Network Node; it is reserved if the local node is any other type.

If the parameter *adj_cp_type* is set to `NETWORK_NODE`, `END_NODE`, `APPN_NODE`, or `BACK_LEVEL_LEN_NODE`, this parameter defines whether the link is an uplink or a downlink. Possible values are:

UPLINK The link is an uplink.

DOWNLINK
The link is a downlink.

adj_brnn_cp_support
This parameter applies only if the local node is a Branch Network Node and the adjacent node is a network node (the parameter *adj_cp_type* is set

to NETWORK_NODE, or it is set to APPN_NODE and the node type discovered during XID exchange is network node). It is reserved if the local and remote nodes are any other type.

This parameter defines whether the adjacent node can be a Branch Network Node that is acting as a Network Node from the point of view of the local node. Possible values are:

ALLOWED
The adjacent node is allowed (but not required) to be a Branch Network Node.

REQUIRED
The adjacent node must be a Branch Network Node.

PROHIBITED
The adjacent node must not be a Branch Network Node.

dddlu_offline_supported
Specifies whether the local PU should send NMVT (power off) messages to the host. If the host system supports DDDLU (Dynamic Definition of Dependent LUs), Communications Server for Linux sends NMVT (power off) to the host when it has finished using a dynamically defined LU. This allows the host to save resources by removing the definition when it is no longer required.

This parameter is used only if this link is to a host (*solicit_sscp_sessions* is set to YES and *dspu_services* is not set to NONE).

Possible values are:

YES The local PU sends NMVT (power off) messages to the host.

NO The local PU does not send NMVT (power off) messages to the host.

If the host supports DDDLU but does not support the NMVT (power off) message, this parameter must be set to NO.

initially_active
Specifies whether this LS is automatically started when the node is started. Possible values are:

YES The LS is automatically started when the node is started.

NO The LS is not automatically started; it must be manually started.

restart_on_normal_deact
Specifies whether Communications Server for Linux should attempt to reactivate the LS if it is deactivated normally by the remote system. Possible values are:

YES If the remote system deactivates the LS normally, Communications Server for Linux attempts to reactivate it, using the same retry timer and count values as for reactivating a failed LS (the *react_timer* and *react_timer_retry* parameters above).

NO If the remote system deactivates the LS normally, Communications Server for Linux does not attempt to reactivate it.

If the LS is a host link (specified by the *adj_cp_type* parameter), or is automatically started when the node is started (the *initially_active* parameter is set to YES), this parameter is ignored; Communications Server

for Linux always attempts to reactivate the LS if it is deactivated normally by the remote system (unless *react_timer_retry* is zero).

react_timer
Reactivation timer for reactivating a failed LS. If the *react_timer_retry* parameter is a nonzero value (to specify that Communications Server for Linux should retry activating the LS if it fails), this parameter specifies the time in seconds between retries. When the LS fails, or when an attempt to reactivate it fails, Communications Server for Linux waits for the specified time before retrying the activation. If *react_timer_retry* is `0` (zero), this parameter is ignored.

react_timer_retry
Retry count for reactivating a failed LS. This parameter is used to specify whether Communications Server for Linux should attempt to reactivate the LS if it fails while in use (or if an attempt to start the LS fails).

A value of `0` (zero) indicates that Communications Server for Linux does not attempt to reactivate the LS. A value of `65,535` indicates that Communications Server for Linux retries indefinitely until the LS is reactivated.

Communications Server for Linux waits for the time specified by the *react_timer* parameter between successive retries. If the LS is not successfully reactivated by the end of the retry count, or if **stop_ls** is issued while Communications Server for Linux is retrying the activation, no further retries are made; the LS remains inactive unless **start_ls** is issued for it.

If the *auto_act_supp* parameter is set to `YES`, the *react_timer* and *react_timer_retry* parameters are ignored; if the link fails, Communications Server for Linux does not attempt to reactivate it until the user application that was using the session attempts to restart the session.

address For an SDLC link station, this parameter identifies the address of the secondary station on this LS.

The value of this parameter depends on how the port that owns this LS is configured, as follows:

- If the port is used only for incoming calls (*out_link_act_lim* on **define_sdlc_port** is `0`), this parameter is reserved.
- If the port is switched primary and is used for outgoing calls (*port_type* is `PORT_SWITCHED`, *ls_role* is `LS_PRI`, and *out_link_act_lim* on **define_sdlc_port** is a nonzero value), this parameter is set to either `0xFF` to accept whatever address is configured at the secondary station, or set it to a 1-byte value in the range `0x01`–`0xFE` (this value must match the value configured at the secondary station).
- For all other port configurations, this parameter is set to a 1-byte value in the range `0x01`–`0xFE` to identify the link station. If the port is primary multi-drop (*ls_role* on **define_sdlc_port** is `LS_PRI` and *tot_link_act_lim* is greater than 1), this address must be different for each LS on the port.

address For a QLLC link station, this parameter identifies the destination address of the remote link station. This parameter is used only for SVC outgoing calls (defined by the *vc_type* parameter on this command and by the link activation limit parameters on **define_qllc_port**); it is ignored for incoming calls or for PVC.

The address is a string of 1–14 characters. The address is in X.25 (1980) format; later address formats are not supported.

mac_address
Token Ring / Ethernet : MAC address of the link station on the adjacent node.

If this parameter is not specified, the LS is a non-selective listening LS (one that can be used only for incoming calls, but can have LUs defined on it to support dependent LU traffic). The LS can be used to receive incoming calls from any remote link station, but cannot be used for outgoing calls.

If the local and adjacent nodes are on LANs of different types (one Ethernet, the other Token Ring) connected by a bridge, you will probably need to reverse the bit order of the bytes in the MAC address. For more information about bit ordering in MAC addresses, see "define_tr_ls, define_ethernet_ls" on page 209. If the two nodes are on the same LAN or on LANs of the same type connected by a bridge, no change is required.

lsap_address
Token Ring / Ethernet : Local SAP address of the link station on the adjacent node.

determined_ip_address
Enterprise Extender (HPR/IP): IP address of the link station on the adjacent node. This is a dotted-decimal IPv4 address (such as `193.1.11.100`) or an IPv6 colon-hexadecimal address (such as `2001:0db8:0000:0000:0000:0000:1428:57ab` or `2001:db8::1428:57ab`). If the LS is inactive, the address appears as all zeros.

For details of the remaining parameters, see "define_tr_ls, define_ethernet_ls" on page 209, "define_sdlc_ls" on page 162, "define_qllc_ls" on page 135, "define_ip_ls" on page 67.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
`PARAMETER_CHECK`

secondary_rc
Possible values are:

`INVALID_LINK_NAME`
The *list_options* parameter was set to `LIST_INCLUSIVE` to list all entries starting from the supplied name, but the value specified in the *ls_name* parameter was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_ls_routing

The **query_ls_routing** command returns information for local LUs about the location of a partner LU using a link station. If information is requested about more than one local LU, the information is ordered by local LU name and then by the partner LU names associated with each local LU name. Wildcard partner LU names can be interspersed with entries that do not contain wildcards.

Supplied Parameters

Parameter name	Type	Length	Default
[query_ls_routing]			
num_entries	decimal		1
list_options	constant		LIST_INCLUSIVE
lu_name	character	8	
fq_partner_lu	character	17	
wildcard_fqplu	constant		NO

Supplied parameters are:

num_entries
: Maximum number of LS routing entries for which data should be returned. You can specify 1 to return data for a specific LS routing entry, a number greater than 1 to return data for multiple LS routing entries, or 0 (zero) to return data for all LS routing entries.

list_options
: The position in the list of LS routing entries from which Communications Server for Linux begins to return data.

  Specify one of the following values:

  FIRST_IN_LIST
  : Start at the first entry in the list.

  LIST_INCLUSIVE
  : Start at the entry specified by the combination of the *lu_name* and *fq_partner_lu* parameters.

  LIST_FROM_NEXT
  : Start at the entry immediately following the entry specified by the combination of the *lu_name*, *fq_partner_lu*, and *wildcard_fqplu* parameters.

lu_name
: Name of the local LU for which routing data is to be returned. This name is an 8-byte character string. This parameter is ignored if *list_options* is set to FIRST_IN_LIST.

fq_partner_lu
: Fully qualified name of the partner LU for which routing data is to be returned. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character partner LU name. This parameter is ignored if *list_options* is set to FIRST_IN_LIST.

  If this parameter is set to binary zeros and *list_options* is set to LIST_FROM_NEXT, the returned list starts at the first partner LU name for the LU identified by the *lu_name* parameter.

wildcard_fqplu
: Wildcard partner LU flag indicating whether the *fq_partner_lu* parameter contains a full or partial wildcard. This flag is used only to identify the

first record to return. It cannot be used to specify that only entries matching the wildcard specification are to be returned. Possible values are:

YES The *fq_partner_lu* parameter contains a wildcard entry.

NO The *fq_partner_lu* parameter does not contain a wildcard entry.

Returned Parameters

Parameter name	Type	Length
lu_name	character	
fq_partner_lu	character	
wildcard_fqplu	character	
ls_name	character	

If the command executes successfully, the following parameters are returned:

lu_name
Name of the local LU.

fq_partner_lu
Fully qualified name of the partner LU.

wildcard_fqplu
Flag indicating whether the *fq_partner_lu* parameter contains a full or partial wildcard. Possible values are:

YES The *fq_partner_lu* parameter contains a full or partial wildcard.

NO The *fq_partner_lu* parameter does not contain a full or partial wildcard.

ls_name
Name of the link station used for sessions between the LU specified in the *lu_name* parameter and the partner LU specified in the *fq_plu_name* parameter.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_LU_NAME
The *list_options* parameter was set to LIST_INCLUSIVE, but the value specified for the *lu_name* parameter did not match any existing LS routing data record.

INVALID_PARTNER_LU_NAME
The *list_options* parameter was set to LIST_INCLUSIVE, but the value specified by the *fq_partner_lu* parameter did not match any existing LS routing data record for the specified partner LU name.

INVALID_WILDCARD_NAME
The *wildcard_fqplu* parameter was set to YES, but the *fq_partner_lu* parameter was not a valid wildcard name.

State Check
No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions
Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_lu_0_to_3

The **query_lu_0_to_3** command returns information about local LUs of type 0, 1, 2, or 3. This information is structured as determined data (data gathered dynamically during execution, returned only if the node is active) and defined data (the data supplied on the **define_lu_0_to_3** command).

This command can be used to obtain summary or detailed information about a specific LU or about multiple LUs, depending on the options used. The detailed information returned varies slightly according to the type of application that is using the LU, as shown in "Returned Parameters: Detailed Information" on page 399.

Supplied Parameters

Parameter name [query_lu_0_to_3]	Type	Length	Default
num_entries	decimal		1
list_options	constant		SUMMARY + LIST_INCLUSIVE
pu_name	character	8	(null string)
lu_name	character	8	(null string)
host_attachment	constant		NONE

Supplied parameters are:

num_entries
Maximum number of LUs for which data should be returned. You can specify 1 to return data for a specific LU, a number greater than 1 to return data for multiple LUs, or 0 (zero) to return data for all LUs.

list_options
The level of information required for each entry and the position in the list of LUs from which Communications Server for Linux begins to return data.

Specify the level of information required with one of the following values:

SUMMARY
Summary information only

DETAIL Detailed information

Use a + character to combine this value with one of the following values:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the *lu_name* parameter

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *lu_name* parameter

pu_name
PU name for which LU information is required. To list only information about LUs associated with a specific PU, specify the PU name. To obtain a complete list for all PUs, do not specify this parameter.

lu_name
Name of the local LU. This name is a type-A character string starting with a letter. This parameter is ignored if *list_options* is set to `FIRST_IN_LIST`.

host_attachment
Host attachment filter. If the command is issued to a running node, this parameter specifies whether to filter the returned information by whether the LUs are attached to the host directly or using DLUR or PU Concentration. Possible values are:

DIRECT_ATTACHED
Return information only about LUs directly attached to the host system.

DLUR_ATTACHED
Return information only about LUs supported by DLUR on the local node.

DLUR Return information only on LUs supported by passthrough DLUR from a downstream node. This option is valid only if the local node is a Network Node.

PU_CONCENTRATION
Return information only on LUs supported by SNA gateway from a downstream node.

NONE Return information about all LUs regardless of host attachment.

If the node is not running, this parameter is ignored; Communications Server for Linux returns information about all LUs regardless of host attachment.

Returned Parameters: Summary Information

Parameter name	Type	Length
pu_name	character	8
lu_name	character	8
description	character	31
nau_address	decimal	
lu_sscp_sess_active	constant	
appl_conn_active	constant	
plu_sess_active	constant	
host_attachment	constant	

If the command executes successfully and you specified `SUMMARY` as the *list_options* parameter value, the following parameters are returned:

pu_name
Name of the local PU used by the LU.

lu_name
Name of the local LU.

description
A text string describing the LU, as specified in the definition of the LU.

nau_address
Network accessible unit address of the LU. This address is in the range 1–255.

lu_sscp_sess_active
Specifies whether the LU-SSCP session is active. Possible values are:

YES The session is active.

NO The session is inactive.

appl_conn_active
Specifies whether an application is using the LU. Possible values are:

YES An application is using the LU.

NO No application is using the LU.

plu_sess_active
Specifies whether the PLU-SLU session is active. Possible values are:

YES The session is active.

NO The session is inactive.

host_attachment
LU host attachment type.

When the command is issued to a running node, this parameter takes one of the following values:

DIRECT_ATTACHED
LU is directly attached to the host system.

DLUR_ATTACHED
LU is supported by DLUR on the local node.

DLUR LU is supported by passthrough DLUR from a downstream node.

PU_CONCENTRATION
LU is supported by SNA gateway from a downstream node.

Returned Parameters: Detailed Information

The detailed information that is returned varies slightly according to the type of application that is using the LU. "Returned Parameters for All Application Types" shows the parameters that are returned in every case. "Additional Returned Parameters for an LU Used by 3270" on page 404 through "Returned Parameters for an LU Used by an LUA Application" on page 407 show the returned parameters that depend on how the LU is being used.

Returned Parameters for All Application Types

The following parameters are returned for any LU that has been defined with a **define_lu_0_to_3** command:

Parameter name	Type	Length
lu_name	character	8
lu_sscp_sess_active	constant	
appl_conn_active	constant	
plu_sess_active	constant	
host_attachment	constant	
lu_sscp_rcv_ru_size	decimal	
lu_sscp_send_ru_size	decimal	
lu_sscp_max_send_btu_size	decimal	
lu_sscp_max_rcv_btu_size	decimal	
lu_sscp_max_send_pac_win	decimal	
lu_sscp_cur_send_pac_win	decimal	

```
lu_sscp_max_rcv_pac_win      decimal
lu_sscp_cur_rcv_pac_win      decimal
lu_sscp_send_data_frames     decimal
lu_sscp_send_fmd_data_frames decimal
lu_sscp_send_data_bytes      decimal
lu_sscp_rcv_data_frames      decimal
lu_sscp_rcv_fmd_data_frames  decimal
lu_sscp_rcv_data_bytes       decimal
lu_sscp_sidh                 hex number
lu_sscp_sidl                 hex number
lu_sscp_odai                 constant
lu_sscp_ls_name              character    8
lu_sscp_pacing_type          constant
plu_rcv_ru_size              decimal
plu_send_ru_size             decimal
plu_max_send_btu_size        decimal
plu_max_rcv_btu_size         decimal
plu_max_send_pac_win         decimal
plu_cur_send_pac_win         decimal
plu_max_rcv_pac_win          decimal
plu_cur_rcv_pac_win          decimal
plu_send_data_frames         decimal
plu_send_fmd_data_frames     decimal
plu_send_data_bytes          decimal
plu_rcv_data_frames          decimal
plu_rcv_fmd_data_frames      decimal
plu_rcv_data_bytes           decimal
plu_sidh                     hex number
plu_sidl                     hex number
plu_odai                     constant
plu_ls_name                  character    8
plu_pacing_type              constant
plu_name                     character    8
active_sscp_id               hex array    8
compression                  constant
session_id                   hex array    8
description                  character    31
nau_address                  decimal
pool_name                    character    8
pu_name                      character    8
priority                     constant
lu_model                     constant
allowed_sscp_id              hex array    8
timeout                      decimal
term_method                  constant
disconnect_on_unbind         constant
```

If the command executes successfully and you specified DETAIL as the *list_options* parameter value, the following parameters are returned:

lu_name
Name of the local LU.

lu_sscp_sess_active
Specifies whether the LU-SSCP session is active. Possible values are:

YES The session is active.

NO The session is inactive.

appl_conn_active
Specifies whether an application is using the LU. Possible values are:

YES An application is using the LU.

NO No application is using the LU.

plu_sess_active
Specifies whether the PLU-SLU session is active. Possible values are:

YES The session is active.

NO The session is inactive.

host_attachment
LU host attachment type.

When the command is issued to a running node, this parameter takes one of the following values:

DIRECT_ATTACHED
LU is directly attached to the host system.

DLUR_ATTACHED
LU is supported by DLUR on the local node.

DLUR LU is supported by passthrough DLUR from a downstream node.

PU_CONCENTRATION
LU is supported by SNA gateway from a downstream node.

For each of the two sessions (LU-SSCP session and PLU-SLU session), the following parameters are included. The parameter names must begin with either *lu_sscp_* or *plu_* to distinguish between the two session types:

rcv_ru_size
Maximum RU size that can be received. (In the LU-SSCP session statistics, this parameter is reserved.)

send_ru_size
Maximum RU size that can be sent. (In the LU-SSCP session statistics, this parameter is reserved.)

max_send_btu_size
Maximum BTU size that can be sent.

max_rcv_btu_size
Maximum BTU size that can be received.

max_send_pac_win
Maximum size of the send pacing window on this session. (In the LU-SSCP session statistics, this parameter is reserved.)

cur_send_pac_win
Current size of the send pacing window on this session. (In the LU-SSCP session statistics, this parameter is reserved.)

max_rcv_pac_win
Maximum size of the receive pacing window on this session. (In the LU-SSCP session statistics, this parameter is reserved.)

cur_rcv_pac_win
Current size of the receive pacing window on this session. (In the LU-SSCP session statistics, this parameter is reserved.)

send_data_frames
Number of normal flow data frames sent.

send_fmd_data_frames
Number of normal flow FMD data frames sent.

send_data_bytes
Number of normal flow data bytes sent.

rcv_data_frames
Number of normal flow data frames received.

rcv_fmd_data_frames
Number of normal flow FMD data frames received.

rcv_data_bytes
Number of normal flow data bytes received.

The following three parameters identify the Local Form Session Identifier (LFSID):

sidh Session ID high byte.

sidl Session ID low byte.

odai Origin Destination Assignor Indicator. Possible values are:

YES The BIND sender is the node containing the secondary link station.

NO The BIND sender is the node containing the primary link station.

ls_name
Link station name associated with statistics.

pacing_type
Receive pacing type in use on the PLU-SLU session. Possible values are:
```
NONE
PACING_FIXED
```

The following parameters are not distinguished by session type:

plu_name
Name of the primary LU. This parameter is reserved if the PLU-SLU session is inactive.

active_sscp_id
The SSCP ID received in the ACTPU for the PU used by this LU. If *lu_sscp_sess_active* is `NO`, this parameter will be all zeros.

compression
Compression level in use on the PLU-SLU session, if any. Possible values are:

NO Data flowing on the PLU-SLU session is not compressed by Communications Server for Linux, or there is no active PLU-SLU session.

YES Communications Server for Linux performs compression and decompression on PLU-SLU session data. RLE compression is used on data flowing upstream to the primary LU, and LZ9 compression is used on data flowing downstream from the primary LU.

PASSTHRU
Compression on this session is performed by the session endpoints (the host LU and the local application or downstream LU), and not by Communications Server for Linux.

session_id
Eight-byte internal identifier of the PLU-SLU session.

description
A text string describing the LU, as specified in the definition of the LU.

nau_address
Network accessible unit address of the LU, in the range 1–255.

pool_name
Name of the LU pool to which this LU belongs. If the LU does not belong to a pool, this parameter is not used.

pu_name
Name of the PU that this LU uses.

priority
LU priority when sending to the host. Possible values are:

NETWORK
The LU has priority on the network.

HIGH High priority is given to the LU.

MEDIUM Medium priority is given to the LU.

LOW Low priority is given to the LU.

lu_model
Type of the LU. Possible values are:

3270_DISPLAY_MODEL_2
LU type is a 3270 display model 2.

3270_DISPLAY_MODEL_3
LU type is a 3270 display model 3.

3270_DISPLAY_MODEL_4
LU type is a 3270 display model 4.

3270_DISPLAY_MODEL_5
LU type is a 3270 display model 5.

PRINTER
LU type is a printer.

SCS_PRINTER
LU type is an SCS printer.

UNKNOWN
LU type is unknown.

allowed_sscp_id
Specifies the ID of the SSCP permitted to activate this LU. If this parameter is set to binary zeros, the LU may be activated by any SSCP.

timeout
Timeout for the LU, specified in seconds. If a timeout is supplied and the user of the LU specified allow_timeout on the OPEN_LU_SSCP_SEC_RQ (or, in the case of SNA gateway, on the downstream LU definition), then the LU will be deactivated after the PLU-SLU session is left inactive for this period and one of the following conditions applies:

- The session passes over a limited resource link.
- Another application wishes to use the LU before the session is used again.

If the timeout is set to zero, the LU will not be deactivated.

term_method
This parameter specifies how Communications Server for Linux should attempt to end a PLU-SLU session to a host from this LU. Possible values are:

USE_NODE_DEFAULT
Use the node's default termination method, specified by the *send_term_self* parameter on **define_node**.

SEND_UNBIND
End the session by sending an UNBIND.

SEND_TERM_SELF
End the session by sending a TERM_SELF.

disconnect_on_unbind
This parameter applies only when this LU is being used by a TN3270 client. It specifies whether to end the session when the host sends an UNBIND instead of displaying the VTAM MSG10 or returning to a host session manager. Possible values are:

YES End the session if the host sends an UNBIND that is not type 2 (BIND forthcoming).

NO Do not end the session if the host sends an UNBIND.

Additional Returned Parameters for an LU Used by 3270

The detailed information returned contains the following parameters in addition to those shown in "Returned Parameters: Detailed Information" on page 399:

```
app_type                  constant
user_name                 character    32
system_name               character    32
user_pid                  decimal
user_type                 constant
user_uid                  decimal
user_gid                  decimal
user_gname                character    32
description               character    31
```

The following parameters are returned:

app_type
Type of application using this LU. This parameter is set to `FMI_APPLICATION`.

user_name
The user name with which the 3270 emulation program is running.

system_name
The computer name on which the program is running.

user_pid
The process ID of the program using the LU.

user_type
Type of session requested by the program using this LU. Possible values are:

3270_DISPLAY_MODEL_2
The program requested a 3270 display model 2 session.

3270_DISPLAY_MODEL_3
The program requested a 3270 display model 3 session.

3270_DISPLAY_MODEL_4
The program requested a 3270 display model 4 session.

3270_DISPLAY_MODEL_5
The program requested a 3270 display model 5 session.

PRINTER
The program requested a printer session.

SCS_PRINTER
The program requested an SCS printer session.

UNKNOWN
The session type is unknown. This value is returned only if the session is inactive.

user_uid
The user ID with which the program is running.

user_gid
The group ID with which the program is running.

user_gname
The group name with which the program is running.

Returned Parameters for an LU used by SNA Gateway

The detailed information returned contains the following parameters in addition to those shown in "Returned Parameters: Detailed Information" on page 399:

```
Parameter name          Type          Length
app_type                constant
pu_conc_downstream_lu   character     8
```

app_type
Type of application using this LU. This parameter is set to PU_CONCENTRATION.

pu_conc_downstream_lu
The name of the downstream LU associated with this LU.

Returned Parameters for an LU Used by a TN Server

The detailed information returned contains the following parameters in addition to those shown in "Returned Parameters: Detailed Information" on page 399:

```
Parameter name        Type          Length
app_type              constant
user_ip_address       character     39
port_number           decimal
cb_number             decimal
cfg_default           constant
cfg_address           character     64
cfg_format            constant
tn3270_level          constant
lu_select             constant
request_lu_name       character     8
cipher_spec           constant
```

app_type
Type of application using this LU. This parameter is set to TN_SERVER.

user_ip_address
The IP address of the computer where the TN3270 program is running. This is a null-terminated ASCII string, which can be either of the following.

- An IPv4 dotted-decimal address (such as 193.1.11.100).

- An IPv6 colon-hexadecimal address (such as `2001:0db8:0000:0000:0000:0000:1428:57ab` or `2001:db8::1428:57ab`).

port_number
The TCP/IP port number that the TN3270 program uses to access TN server.

cb_number
TN server control block number.

cfg_default
Specifies whether the TN3270 program is using an explicitly defined TN server user record or is using the configured default record. For more information about configuring a default TN server user record, see "define_tn3270_access" on page 186. Possible values are:

YES The program is using the default record. The *cfg_address* and *cfg_format* parameters are reserved.

NO The program is using an explicitly defined record.

cfg_address
The TCP/IP address of the computer on which the TN3270 program runs, as defined in the configuration record that this user is using.

The address may be specified as an IPv4 dotted-decimal address (such as 193.1.11.100), an IPv6 colon-hexadecimal address (such as `2001:0db8:0000:0000:0000:0000:1428:57ab` or `2001:db8::1428:57ab`), a name (such as `newbox.this.co.uk`), or an alias (such as `newbox`); the format is indicated by the *cfg_format* parameter.

cfg_format
Specifies the format of the *cfg_address* parameter. Possible values are:

IP_ADDRESS
IP address

FULLY_QUALIFIED_NAME
Alias or fully qualified name

tn3270_level
Level of TN3270 support. Possible values are:

LEVEL_TN3270
TN3270E protocols are disabled.

LEVEL_TN3270E
TN3270E protocols are enabled.

lu_select
Method of LU selection. Possible values are:

GENERIC_LU
This LU can be used by any TN3270 program that requests a generic display or printer LU.

SPECIFIC_LU
This LU can be used only by a TN3270 program that names this LU specifically.

ASSOCIATED_LU
This LU is a printer LU that has been associated with a display LU

by a **define_tn3270_association** command, or a display LU that has been associated with a printer LU by a **define_tn3270_association** command.

request_lu_name
Requested LU name or associated display LU name.

cipher_spec
Indicates the type of SSL security and the encryption level in use for this session. Possible values are:

SSL_NO_SSL
SSL is not being used.

SSL_NULL_MD5
Certificates are exchanged, but no encryption is used.

SSL_NULL_SHA
Certificates are exchanged, but no encryption is used.

SSL_RC4_MD5_EXPORT
40-bit encryption.

SSL_RC2_MD5_EXPORT
40-bit encryption.

SSL_DES_SHA_EXPORT
56-bit encryption.

SSL_RC4_MD5_US
128-bit encryption.

SSL_RC4_SHA_US
128-bit encryption.

SSL_3DES_SHA_US
Triple–DES (168-bit) encryption.

Returned Parameters for an LU Used by an LUA Application

The detailed information returned contains the following parameters in addition to those shown in "Returned Parameters: Detailed Information" on page 399:

```
Parameter name              Type         Length
app_type                    constant
user_ip_address             character    39
user_host_address           character    255
```

app_type
Type of application using this LU. This parameter is set to `LUA_APPLICATION`.

user_ip_address
The IP address of the computer (client or server) where the LUA application is running. This can be either of the following.

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

user_host_address
The name of the computer (client or server) where the LUA application is running. This is an IP hostname (such as `newbox.this.co.uk`).

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_LU_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *lu_name* parameter value was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

query_lu_lu_password

The **query_lu_lu_password** command returns information about passwords used for session-level security verification between a local LU and a partner LU. The command can be used to obtain information about the password for a specific partner LU or about passwords for multiple partner LUs, depending on the options used.

Supplied Parameters

Parameter name [query_lu_lu_password]	Type	Length	Default
num_entries	decimal		1
list_options	constant		LIST_INCLUSIVE
lu_name	character	8	(null string)
lu_alias	character	8	(null string)
plu_alias	character	8	(null string)
fqplu_name	character	17	(null string)

Supplied parameters are:

num_entries
Maximum number of LUs for which data should be returned. You can specify 1 to return data for a specific LU, a number greater than 1 to return data for multiple LUs, or 0 (zero) to return data for all LUs.

list_options
The position in the list of LUs from which Communications Server for Linux begins to return data.

Specify one of the following values:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the *plu_alias* or *fqplu_name* parameter

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *plu_alias* or *fqplu_name* parameter

lu_name
LU name. This name is a type-A character string. To indicate that the LU is identified by its LU alias instead of its LU name, do not specify this parameter, and specify the LU alias in the *lu_alias* parameter.

lu_alias
Locally defined LU alias. This alias is an 8-byte string of locally displayable characters. This parameter is used only if *lu_name* is not specified. To indicate the LU associated with the CP (the default LU), do not specify either *lu_name* or *lu_alias*.

plu_alias
Partner LU alias. This alias is an 8-byte string of locally displayable characters. If *list_options* is set to FIRST_IN_LIST, this parameter is ignored; otherwise, you must specify either the LU alias or the fully qualified LU name for the partner LU. To indicate that the partner LU is identified by its fully qualified LU name instead of its LU alias, do not specify this parameter, and specify the LU alias in the *fqplu_name* parameter.

fqplu_name
Fully qualified name of the partner LU. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character partner LU name. For the network name and partner LU name, use only letters, digits 0–9, and special characters $, #, and @.

The name can be used as the partner LU name for which information is required or as an index into the list of LUs. If *list_options* is set to FIRST_IN_LIST, this parameter is ignored; otherwise, you must specify either the LU alias or the fully qualified LU name for the partner LU. This parameter is used only if the *plu_alias* parameter is not specified; it is ignored if *plu_alias* is specified.

Returned Parameters

Parameter name	Type	Length
plu_alias	character	8
fqplu_name	character	17
description	character	31
protocol_defined	constant	1
protocol_in_use	constant	1

If the command executes successfully, the following parameters are returned:

plu_alias
Partner LU alias.

fqplu_name
A 17-byte fully qualified network name of the partner LU.

description
A text string describing the LU-LU password, as specified in the definition of the password.

protocol_defined
Requested LU-LU verification protocol defined for use with this partner LU. Possible values are:

BASIC Basic security protocols requested.

ENHANCED
Enhanced security protocols requested.

EITHER Basic or enhanced security is accepted.

protocol_in_use
LU-LU verification protocol in use with this partner LU. Possible values are:

BASIC Basic security protocols requested.

ENHANCED
Enhanced security protocols requested.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_LU_ALIAS
The supplied *lu_alias* parameter did not match the alias of any configured LU.

INVALID_LU_NAME
The supplied *lu_name* parameter did not match the name of any configured LU.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_lu_pool

The **query_lu_pool** command returns information about LU pools and the LUs that belong to them. If the node is active, it also returns status information indicating whether sessions for the LUs are active.

This command can be used to obtain information about a specific LU or pool or about multiple LUs or pools, depending on the options used.

Supplied Parameters

```
Parameter name          Type          Length     Default
[query_lu_pool]
num_entries             decimal                  1
list_options            constant                 SUMMARY + LIST_INCLUSIVE
pool_name               character     8          (null string)
lu_name                 character     8          (null string)
```

Supplied parameters are:

num_entries
: Maximum number of LU pools or LUs in a pool (depending on level of information to be returned) for which data should be returned. You can specify 1 to return data for a specific entry, a number greater than 1 to return data for multiple entries, or 0 (zero) to return data for all entries.

  If *list_options* is set to SUMMARY, each entry is a single LU pool; if *list_options* is set to DETAIL, each entry is an LU in a pool (or an entry indicating an empty LU pool).

list_options
: The level of information required for each entry and the position in the list of entries from which Communications Server for Linux begins to return data. The list is ordered by *pool_name* and then by *lu_name* if detailed information is being returned.

  Specify the level of information required with one of the following values:

  SUMMARY
  : Summary information only

  DETAIL Detailed information lists individual LUs in LU pools

  Use a + character to combine this value with one of the following values:

  FIRST_IN_LIST
  : Start at the first entry in the list

  LIST_INCLUSIVE
  : Start at the entry specified by the combination of the *pool_name* and *lu_name* parameters

  LIST_FROM_NEXT
  : Start at the entry immediately following the entry specified by the combination of the *pool_name* and *lu_name* parameters

pool_name
: Name of the LU pool for which information is required. This parameter is an 8-byte character string. It is ignored if *list_options* is set to FIRST_IN_LIST.

lu_name
: LU name for which information is required. This parameter is an 8-byte character string. It is ignored if *list_options* is set to SUMMARY or FIRST_IN_LIST.

  To obtain information about all LUs in a pool, set *pool_name* to the name of the pool, set *num_entries* to 0 (zero), and do not specify *lu_name*.

Returned Parameters: Summary Information

Parameter name	Type	Length
pool_name	character	8
description	character	31
num_active_lus	decimal	
num_avail_lus	decimal	

If the command executes successfully and you specified SUMMARY as the *list_options* parameter value, the following parameters are returned:

pool_name
Name of the LU pool.

description
A text string describing the LU pool, as specified in the definition of the pool.

num_active_lus
Number of LUs in the pool that are active.

num_avail_lus
Number of LUs in the pool that are available for activation by a forced open request. It includes all LUs whose PU is active or whose host link can be auto-activated, and whose connection is free.

This count does not take account of the LU *model_type*, *model_name* and the DDDLU support of the PU. If the open request specifies a particular value for *model_type*, some LUs that are included in this count may not be available because they do not have the correct model type.

Returned Parameters: Detailed Information

Parameter name	Type	Length
pool_name	character	8
description	character	31
lu_name	character	8
lu_sscp_sess_active	constant	
appl_conn_active	constant	
plu_sess_active	constant	

If the command executes successfully and you specified DETAIL as the *list_options* parameter value, the following parameters are returned:

pool_name
Name of LU pool to which the LU belongs.

description
A text string describing the LU pool, as specified in the definition of the pool.

lu_name
LU name. If a single entry is returned for a particular pool name with no LU name, this indicates that the LU pool is empty.

lu_sscp_sess_active
Specifies whether the LU-SSCP session is active. Possible values are:

YES The session is active.

NO The session is inactive.

appl_conn_active
Specifies whether an application is using the LU. Possible values are:

YES An application is using the LU.

NO No application is using the LU.

plu_sess_active
Specifies whether the PLU-SLU session is active. Possible values are:

YES The session is active.

NO The session is inactive.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_LU_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *lu_name* parameter value was not valid.

INVALID_POOL_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *pool_name* parameter value was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_lu62_timeout

The **query_lu62_timeout** command returns information about the definition of an LU type 6.2 session timeout that was defined previously with a **define_lu62_timeout** command.

The information is returned as a list. To obtain information about a specific timeout, or about several timeout values, specify values for the *resource_type* and *resource_name* parameters. If the *list_options* parameter is set to FIRST_IN_LIST, the *resource_type* and *resource_name* parameters are ignored. The returned list is ordered on *resource_type* and then on *resource_name*.

For *resource_type*, the ordering is:

1. Global timeouts

2. Local LU timeouts
3. Partner LU timeouts
4. Mode timeouts

For *resource_name*, the ordering is by:

1. Name length
2. By ASCII lexicographical ordering for names of the same length

If the *list_options* parameter is set to LIST_FROM_NEXT, the returned list starts for the next entry according to the defined ordering (whether or not the specified entry exists).

Supplied Parameters

Parameter name [query_lu62_timeout]	Type	Length	Default
num_entries	decimal	1	
list_options	constant		LIST_INCLUSIVE
resource_type	constant		GLOBAL_TIMEOUT
resource_name	character	17	(null string)

Supplied parameters are:

num_entries
: Maximum number of entries for which data should be returned. You can specify 1 to return data for a specific entry, a number greater than 1 to return data for multiple entries, or 0 (zero) to return data for all entries.

list_options
: The position in the list of entries from which Communications Server for Linux begins to return data. The list is ordered by *resource_type* in the order GLOBAL_TIMEOUT, LOCAL_LU_TIMEOUT, PARTNER_LU_TIMEOUT, MODE_TIMEOUT, then by *resource_name* in order of the name length, then by lexicographical ordering for names of the same length.

  Possible values are:

  FIRST_IN_LIST
  : Start at the first entry in the list

  LIST_INCLUSIVE
  : Start at the entry specified by the combination of the *resource_type* and *resource_name* parameters

  LIST_FROM_NEXT
  : Start at the entry immediately following the entry specified by the combination of the *resource_type* and *resource_name* parameters

resource_type
: Specifies the type of timeout being queried. This parameter is ignored if *list_options* is set to FIRST_IN_LIST.

  Possible values are:

  GLOBAL_TIMEOUT
  : Timeout applies to all LU 6.2 sessions for the local node.

  LOCAL_LU_TIMEOUT
  : Timeout applies to all LU 6.2 sessions for the local LU specified in the *resource_name* parameter.

PARTNER_LU_TIMEOUT
Timeout applies to all LU 6.2 sessions to the partner LU specified in the *resource_name* parameter.

MODE_TIMEOUT
Timeout applies to all LU 6.2 sessions using the mode specified in the *resource_name* parameter.

resource_name
Name of the resource being queried. This value can be one of the following:

- If *resource_type* is set to GLOBAL_TIMEOUT, do not specify this parameter.
- If *resource_type* is set to LOCAL_LU_TIMEOUT, specify 1–8 type-A characters as a local LU name.
- If *resource_type* is set to PARTNER_LU_TIMEOUT, specify the fully qualified name of the partner LU as follows: 3–17 type-A characters consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character partner LU name.
- If *resource_type* is set to MODE_TIMEOUT, specify 1–8 type-A characters as a mode name.

This parameter is ignored if *list_options* is set to FIRST_IN_LIST.

Returned Parameters

Parameter name	Type	Length
resource_type	constant	
resource_name	character	17
timeout	decimal	

If the command executes successfully, Communications Server for Linux returns the following parameters:

resource_type
The type of the timeout. Possible values are:

GLOBAL_TIMEOUT
Timeout applies to all LU 6.2 sessions for the local node. The *resource_name* parameter is set to all zeros.

LOCAL_LU_TIMEOUT
Timeout applies to all LU 6.2 sessions for the local LU indicated by the *resource_name* parameter.

PARTNER_LU_TIMEOUT
Timeout applies to all LU 6.2 sessions to the partner LU indicated by the *resource_name* parameter.

MODE_TIMEOUT
Timeout applies to all LU 6.2 sessions using the mode indicated by the *resource_name* parameter.

resource_name
Name of the resource. This name is a local LU, a partner LU, or a mode, depending on the value of the *resource_type* parameter. This parameter is set to zeros if *resource_type* is set to GLOBAL_TIMEOUT.

timeout
Timeout period in seconds. A value of 0 (zero) indicates that the session times out immediately after it becomes free.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_RESOURCE_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name and type, but the combination of *resource_type* and *resource_name* did not match any that are configured.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

query_mds_application

The **query_mds_application** command returns a list of applications that have registered for MDS-level messages (by issuing the MS verb REGISTER_MS_APPLICATION). For more information about this MS verb, refer to the *Communications Server for Linux MS Programmer's Guide*. This command can be used to obtain information about a specific application or about multiple applications, depending on the options used.

This command must be issued to a running node.

Supplied Parameters

Parameter name	Type	Length	Default
[query_mds_application]			
num_entries	decimal		1
list_options	constant		LIST_INCLUSIVE
application	character	8	(null string)

Supplied parameters are:

num_entries
Maximum number of applications for which data should be returned. You can specify 1 to return data for a specific application, a number greater than 1 to return data for multiple applications, or 0 (zero) to return data for all applications.

list_options
The position in the list of applications from which Communications Server for Linux begins to return data.

Possible values are:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the *application* parameter

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *application* parameter

application
Application name for which information is required, or the name to be used as an index into the list. This parameter is ignored if *list_options* is set to FIRST_IN_LIST. This name is a type-A character string.

Returned Parameters

Parameter name	Type	Length
application	character	8
max_rcv_size	decimal	

If the command executes successfully, Communications Server for Linux returns the following parameters:

application
Name of registered application.

max_rcv_size
The maximum number of bytes the application can receive in one message (specified when an application registers with MDS). For more information about MDS-level application registration, refer to the *Communications Server for Linux MS Programmer's Guide*.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_APPLICATION_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the value specified in the *application* parameter was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Function Not Supported

If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameters:

primary_rc

FUNCTION_NOT_SUPPORTED
The local node does not support MS network management functions; support is defined by the *mds_supported* parameter in the node definition.

secondary_rc
(This parameter is not used.)

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

query_mds_statistics

The **query_mds_statistics** command returns Management Services statistics. These statistics can be used to gauge the level of MDS routing traffic. The information can also be used to determine the required size of the send alert queue, which is configured as part of the node definition.

This command must be issued to a running node.

Supplied Parameters

```
[query_mds_statistics]
```

No parameters are supplied for this command.

Returned Parameters

Parameter name	Type
alerts_sent	decimal
alert_errors_rcvd	decimal
uncorrelated_alert_errors	decimal
mds_mus_rcvd_local	decimal
mds_mus_rcvd_remote	decimal
mds_mus_delivered_local	decimal
mds_mus_delivered_remote	decimal
parse_errors	decimal
failed_deliveries	decimal
ds_searches_performed	decimal
unverified_errors	decimal

If the command executes successfully, Communications Server for Linux returns the following parameters:

alerts_sent
Number of locally originated alerts sent using the MDS transport system.

alert_errors_rcvd
Number of error messages received by MDS. An error message indicates a delivery failure for a message containing an alert.

uncorrelated_alert_errors
Number of error messages received by MDS. An error message indicates a delivery failure for a message containing an alert. Delivery failure occurs

when the error message cannot be correlated to an alert on the MDS send alert queue. MDS maintains a fixed-size queue where it caches alerts that were sent to the problem determination focal point. When the queue reaches maximum size, the oldest alert is discarded and replaced by the new alert. If a delivery error message is received, MDS attempts to correlate the error message to a cached alert so the alert can be held until the problem determination focal point is restored.

Note: The two counts *alert_errors_rcvd* and *uncorrelated_alert_errors* can be used to check that the size of the send alert queue (specified on the **define_node** command) is appropriate. If the value of *uncorrelated_alert_errors* increases over time, the size of the send alert queue is too small.

mds_mus_rcvd_local
Number of MDS_MUs received from local applications.

mds_mus_rcvd_remote
Number of MDS_MUs received from remote nodes using the MDS_RECEIVE and MSU_HANDLER transaction programs.

mds_mus_delivered_local
Number of MDS_MUs successfully delivered to local applications.

mds_mus_delivered_remote
Number of MDS_MUs successfully delivered to remote nodes using the MDS_SEND transaction program.

parse_errors
Number of MDS_MUs received that contained header format errors.

failed_deliveries
Number of MDS_MUs that this node failed to deliver.

ds_searches_performed
Number of Directory Services searches used to locate the next hop for an MDS_MU. This parameter is significant for network nodes only.

unverified_errors
Number of routing errors caused by using unverified (local Directory Services search) data to determine the next hop for an MDS_MU. Each time an error of this type occurs, Directory Services must repeat the search using either a Central Directory Search or a broadcast search mechanism. This parameter is significant for network nodes only.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Function Not Supported

If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameters:

primary_rc

FUNCTION_NOT_SUPPORTED
The local node does not support MS network management functions; support is defined by the *mds_supported* parameter in the node definition.

secondary_rc
(This parameter is not used.)

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

query_mode

The **query_mode** command returns information about modes that a local LU is using, or has used, with a particular partner LU. The command can be used to obtain information about a specific mode, multiple modes, modes for which sessions are currently active, or all modes that have been used, depending on the options used. The command returns information about current usage of the modes and LUs, not on their definition; use **query_mode_definition** to obtain the definition of the modes and LUs.

This command must be issued to a running node.

Supplied Parameters

Parameter name	Type	Length	Default
[query_mode]			
num_entries	decimal		1
list_options	constant		SUMMARY + LIST_INCLUSIVE
lu_name	character	8	(null string)
lu_alias	character	8	(null string)
plu_alias	character	8	(null string)
fqplu_name	character	17	(null string)
mode_name	character	8	(null string)
active_sessions	constant		NO

Supplied parameters are:

num_entries
Maximum number of modes for which data should be returned. You can specify 1 to return data for a specific mode, a number greater than 1 to return data for multiple modes, or 0 (zero) to return data for all modes.

list_options
The level of information required for each entry and the position in the list of modes from which Communications Server for Linux begins to return data.

Specify the level of information required with one of the following values:

SUMMARY
Summary information only

DETAIL Detailed information

Use a + character to combine this value with one of the following values:

FIRST_IN_LIST
Start at the first entry in the list (the first partner LU for the specified local LU)

LIST_INCLUSIVE
Start at the entry specified by the combination of the *fqplu_name* (or *plu_alias*) and *mode_name* parameters

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the combination of the *fqplu_name* and *mode_name* parameters

For FIRST_IN_LIST, the entry used as the index into the list is defined by the combination of *lu_name* (or *lu_alias*) and *fqplu_name* (or *plu_alias*). If *fqplu_name* or *plu_alias* is not specified, the entry used as the index is *lu_name* (or *lu_alias*).

For LIST_INCLUSIVE or LIST_FROM_NEXT, the entry used as the index into the list is defined by the combination of *lu_name* (or *lu_alias*), *fqplu_name* (or *plu_alias*), and *mode_name* specified.

lu_name
LU name of the local LU, as defined in Communications Server for Linux. This name is a type-A character string. To indicate that the LU is identified by its LU alias instead of its LU name, do not specify this parameter. To specify the LU associated with the local CP (the default LU), do not specify either *lu_name* or *lu_alias*.

lu_alias
Locally defined LU alias. This parameter is used only if *lu_name* is not specified. To indicate the LU associated with the CP (the default LU), do not specify either *lu_name* or *lu_alias*.

plu_alias
Partner LU alias. To indicate that the LU is identified by its LU name rather than its alias, do not specify this parameter.

fqplu_name
Fully qualified name of the partner LU. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by period, followed by a 1–8 character partner LU name. For the network name and partner LU name, use only letters, digits 0–9, and special characters $, #, and @.

This parameter is used only if the *plu_alias* parameter is not specified; it is ignored if *plu_alias* is specified.

mode_name
Mode name that designates the network properties for a group of sessions. This name is a type-A character string. This parameter is ignored if *list_options* is set to FIRST_IN_LIST.

active_sessions
Specifies whether to return information only about modes for which sessions are active or about all modes. Possible values are:

YES Return information only about modes for which sessions are currently active.

NO Return information about all modes for which sessions are active or have been active.

Returned Parameters: Summary Information

Parameter name	Type	Length
mode_name	character	8
description	character	31
sess_limit	decimal	
act_sess_count	decimal	
fqplu_name	character	17

If the command executes successfully and you specified SUMMARY as the *list_options* parameter value, Communications Server for Linux returns the following parameters:

mode_name
Mode name.

description
A text string describing the mode, as specified in the definition of the mode.

sess_limit
Current session limit.

act_sess_count
Total number of active sessions between the specified local LU and partner LU using the mode.

fqplu_name
A 17-byte fully qualified network name of the partner LU.

Returned Parameters: Detailed Information

Parameter name	Type	Length
mode_name	character	8
description	character	31
sess_limit	decimal	
act_sess_count	decimal	
fqplu_name	character	17
min_conwinners_source	decimal	
min_conwinners_target	decimal	
drain_source	constant	
drain_partner	constant	
auto_act	decimal	
act_cw_count	decimal	
act_cl_count	decimal	
sync_level	constant	
default_ru_size	constant	
max_neg_sess_limit	decimal	
max_rcv_ru_size	decimal	
pending_session_count	decimal	
termination_count	decimal	
implicit	constant	

If the command executes successfully and you specified DETAIL as the *list_options* parameter value, Communications Server for Linux returns the following parameters:

mode_name
Mode name.

description
A text string describing the mode, as specified in the definition of the mode.

sess_limit
Current session limit.

act_sess_count
Total number of active sessions between the specified local LU and partner LU using the mode.

fqplu_name
A 17-byte fully qualified network name of the partner LU.

min_conwinners_source
Specifies the minimum number of sessions on which the local (source) LU is the contention winner.

min_conwinners_target
Specifies the minimum number of sessions on which the local LU is the contention loser.

drain_source
Specifies whether the local (source) LU satisfies waiting session requests before deactivating a session when session limits are changed or reset. Possible values are:

YES Waiting session requests are satisfied before sessions are deactivated

NO Waiting session requests are not satisfied before sessions are deactivated

drain_partner
Specifies whether the partner LU satisfies waiting session requests before deactivating a session when session limits are changed or reset. Possible values are:

YES Waiting session requests are satisfied before sessions are deactivated.

NO Waiting session requests are not satisfied before sessions are deactivated.

auto_act
Number of contention winner sessions that are automatically activated following the CNOS exchange with the partner LU.

act_cw_count
Number of active contention winner sessions using this mode.

act_cl_count
Number of active contention loser sessions using this mode.

sync_level
Specifies the synchronization level that the mode supports. Possible values are:

CONFIRM
The mode supports synchronization using the CONFIRM and CONFIRMED verbs.

SYNCPT The mode supports sync point functions.

NONE The mode does not support synchronization.

default_ru_size
Specifies whether the default upper and lower bounds for the maximum RU size is used. Possible values are:

YES Communications Server for Linux ignores the maximum RU size bounds specified in the definition of the mode, and sets the upper bound for the maximum RU size to the default (the largest value that can be accommodated in the link BTU size).

NO Communications Server for Linux uses the maximum RU size bounds specified in the definition of the mode.

max_neg_sess_limit
Specifies the maximum negotiable session limit that a local LU can use with this mode name during its CNOS processing as the target LU.

max_rcv_ru_size
Specifies the maximum RU size received.

pending_session_count
Specifies the number of sessions pending (waiting for session activation).

termination_count
If a previous CNOS command set the mode session limit to 0 (zero), but sessions are still active because conversations were using them or waiting to use them, this parameter specifies the number of sessions that have not yet been deactivated.

implicit
Specifies whether the entry is for an implicit or explicit definition. Possible values are:

YES The entry is for an implicit definition, which was created using the default mode name defined by the **define_defaults** command.

NO The entry is for an explicit definition.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_LU_ALIAS
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *lu_alias* parameter value was not valid.

INVALID_LU_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *lu_name* parameter value was not valid.

INVALID_MODE_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *mode_name* parameter value was not valid.

INVALID_PLU_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but one of the following conditions exists:

- The *fqplu_name* parameter did not match the name of any of this local LU's partners.
- No sessions have been active (since the node was last started) for the specified combination of local LU, partner LU, and mode.

State Check
No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions
Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_mode_definition

The **query_mode_definition** command returns information about modes, including SNA-defined modes. This command can be used to obtain summary or detailed information about a specific mode or about multiple modes, depending on the options used.

This command returns information about the definition of the modes, not about their current usage; use **query_mode** to obtain information about the current usage of a mode by local and partner LUs. Modes are ordered by name length and then by ASCII lexicographical ordering for names of the same length.

This command does not return information about the default COS name that will be used for any unrecognized mode names; use **query_mode_to_cos_mapping** for information about the default COS name.

Supplied Parameters

Parameter name	Type	Length	Default
[query_mode_definition]			
num_entries	decimal		1
list_options	constant		SUMMARY + LIST_INCLUSIVE
mode_name	character	8	(null string)

Supplied parameters are:

num_entries
Maximum number of modes for which data should be returned. You can specify 1 to return data for a specific mode, a number greater than 1 to return data for multiple modes, or 0 (zero) to return data for all modes.

list_options
The level of information required for each entry and the position in the list of modes from which Communications Server for Linux begins to return data.

Specify the level of information required with one of the following values:

SUMMARY
Summary information only

DETAIL Detailed information

Use a + character to combine this value with one of the following values:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the *mode_name* parameter

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *mode_name* parameter

mode_name
Mode name that designates the network properties for a group of sessions. This parameter is ignored if *list_options* is set to FIRST_IN_LIST. This name is a type-A character string.

Returned Parameters: Summary Information

Parameter name	Type	Length
mode_name	character	8
description	character	31

If the command executes successfully and you specified SUMMARY as the *list_options* parameter value, Communications Server for Linux returns the following parameters:

mode_name
Mode name.

description
A text string describing the mode, as specified in the definition of the mode.

Returned Parameters: Detailed Information

Parameter name	Type	Length
mode_name	character	8
description	character	31
max_ru_size_upp	decimal	
receive_pacing_win	decimal	
default_ru_size	constant	
max_neg_sess_lim	decimal	
plu_mode_session_limit	decimal	
min_conwin_src	decimal	
cos_name	character	8
compression	constant	
auto_act	decimal	
min_conloser_src	decimal	
max_ru_size_low	decimal	
max_receive_pacing_win	decimal	
max_compress_level	constant	
max_decompress_level	constant	

If the command executes successfully and you specified DETAIL as the *list_options* parameter value, Communications Server for Linux returns the following parameters:

mode_name
Mode name.

description
A text string describing the mode, as specified in the definition of the mode.

max_ru_size_upp **through** *max_decompress_level*
For information about these parameters, see "define_mode" on page 103.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_MODE_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *mode_name* parameter value was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_mode_to_cos_mapping

The **query_mode_to_cos_mapping** command returns information about the class of service (COS) associated with a particular mode. This command can be used to obtain information about a specific mode or about multiple modes, depending on the options used.

This command must be issued to a running node.

Supplied Parameters

Parameter name	Type	Length	Default
[query_mode_to_cos_mapping]			
num_entries	decimal		1
list_options	constant		LIST_INCLUSIVE
mode_name	character	8	(null string)

Supplied parameters are:

num_entries
Maximum number of modes for which data should be returned. You can specify 1 to return data for a specific mode, a number greater than 1 to return data for multiple modes, or 0 (zero) to return data for all modes.

list_options
The position in the list of modes from which Communications Server for Linux begins to return data.

Possible values are:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the *mode_name* parameter

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *mode_name* parameter

mode_name
The name of the mode for which information is required, or the name to be used as an index into the list. This value is ignored if *list_options* is set to FIRST_IN_LIST. To return information about the default COS that is used for any unrecognized mode names, set this parameter to a pair of angle brackets <> (indicating an empty hexadecimal array).

Returned Parameters

Parameter name	Type	Length
mode_name	character	8
cos_name	character	8

If the command executes successfully, Communications Server for Linux returns the following parameters:

mode_name
Mode name.

cos_name
Class of service name associated with the mode name.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_MODE_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *mode_name* parameter value was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_nmvt_application

The **query_nmvt_application** command returns a list of applications that have registered for NMVT-level messages (by issuing the MS verb REGISTER_NMVT_APPLICATION). This command can be used to obtain information about a specific application or about multiple applications, depending on the options used. For more information about this MS verb, refer to the *Communications Server for Linux MS Programmer's Guide*.

This command must be issued to a running node.

Supplied Parameters

Parameter name	Type	Length	Default
[query_nmvt_application]			
num_entries	decimal		1
list_options	constant		LIST_INCLUSIVE
application	character	8	(null string)

Supplied parameters are:

num_entries
: Maximum number of applications for which data should be returned. You can specify 1 to return data for a specific application, a number greater than 1 to return data for multiple applications, or 0 (zero) to return data for all applications.

list_options
: The position in the list of applications from which Communications Server for Linux begins to return data.

  Possible values are:

  FIRST_IN_LIST
  : Start at the first entry in the list

  LIST_INCLUSIVE
  : Start at the entry specified by the *application* parameter

  LIST_FROM_NEXT
  : Start at the entry immediately following the entry specified by the *application* parameter

application
: Application name for which information is required, or the name to be used as an index into the list of applications. This parameter is ignored if *list_options* is set to FIRST_IN_LIST. The name is a type-A character string.

Returned Parameters

Parameter name	Type	Length
application	character	8
ms_vector_key_type	decimal	
conversion_required	constant	

If the command executes successfully, Communications Server for Linux returns the following parameters:

application
Name of the registered application.

ms_vector_key_type
MS vector key accepted by the application. When the application registers for NMVT messages, it specifies which MS vector keys it accepts. A value of `0xFFFF` indicates that the application has registered for all keys. A value of `0xFFFE` indicates that the application has registered for all SPCF keys.

conversion_required
Specifies whether the registered application requires incoming messages to be converted from NMVT to MDS_MU format. When the application registers for NMVT messages, it specifies whether this conversion is required. Possible values are:

YES Incoming messages are converted to MDS_MU format.

NO Incoming messages are not converted to MDS_MU format.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
`PARAMETER_CHECK`

secondary_rc
Possible values are:

INVALID_APPLICATION_NAME
The *list_options* parameter was set to `LIST_INCLUSIVE` to list all entries starting from the supplied name, but the value specified in the *application* parameter was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_nn_topology_node

Each network node (NN) maintains a network topology database that holds information about all the network nodes, virtual routing nodes (VRNs), and network node-to-network node TGs in the network. The **query_nn_topology_node** command returns information about the network node and VRN entries in this database. This command can be used to obtain summary or detailed information about a specific node or about multiple nodes, depending on the options used.

This command must be issued to a running node. It can be used only if the Communications Server for Linux node is a network node and is not valid if it is an end node or LEN node.

Supplied Parameters

Parameter name	Type	Length	Default
[query_nn_topology_node]			
num_entries	decimal		1
list_options	constant		SUMMARY + LIST_INCLUSIVE
node_name	character	17	(null string)
node_type	constant		LEARN_NODE
frsn	decimal		0

Note: If the *frsn* parameter is set to a nonzero value, then only node entries with FRSNs equal to or greater than the one specified will be returned. If the *frsn* parameter is set to 0 (zero), then all node entries are returned.

Supplied parameters are:

num_entries
Maximum number of nodes for which data should be returned. You can specify 1 to return data for a specific node, a number greater than 1 to return data for multiple nodes, or 0 (zero) to return data for all nodes.

list_options
The position in the list of nodes from which Communications Server for Linux begins to return data and the level of information required for each entry. The list is ordered by *node_name*, then by *node_type* in the order NETWORK_NODE, VRN, and finally in numerical order of *frsn*.

Specify the level of information required with one of the following values:

SUMMARY
Summary information only

DETAIL Detailed information

Use a + character to combine this value with one of the following values:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the combination of the *node_name*, *node_type*, and *frsn* parameters

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the combination of the *node_name*, *node_type*, and *frsn* parameters

node_name
Fully qualified name of the node for which information is required, or the name to be used as an index into the list of nodes. This value is ignored if *list_options* is set to FIRST_IN_LIST. The name is a type-A character string, consisting of 1–8 character network name, followed by a period, followed by a 1–8 character mode name.

node_type
Type of the node. Possible values are:

NETWORK_NODE
Network node (NN)

VRN Virtual routing node (VRN)

LEARN_NODE
Node type is unknown

frsn Flow reduction sequence number. Specify `0` (zero) to return information about all nodes or a nonzero value to return information about nodes with a FRSN greater than or equal to this value.

This parameter can be used to ensure that consistent information is obtained when you need to issue several commands to obtain all required information. Take the following steps:

To Obtain Consistent Information Using the frsn Parameter

1. Issue **query_node** to get the node's current FRSN.
2. Issue as many **query_nn_topology_node** commands as necessary to get all the database entries, with the *frsn* parameter set to `0` (zero).
3. Issue **query_node** again and compare the new FRSN with the one returned in step 1.
4. If the two FRSNs are different, the database has changed. Add `1` to the FRSN obtained in step 1, and issue additional **query_nn_topology_node** commands with the *frsn* parameter set to this new value. These commands return only the entries that have changed.

Returned Parameters: Summary Information

Parameter name	Type	Length
node_name	character	17
node_type	constant	

If the command executes successfully and you specified `SUMMARY` as the *list_options* parameter value, the following parameters are returned:

node_name
Fully qualified name of the node.

node_type
Type of the node. Possible values are:

NETWORK_NODE
Network node (NN)

END_NODE
End node (EN)

VRN Virtual routing node (VRN)

Returned Parameters: Detailed Information

Parameter name	Type	Length
node_name	character	17
node_type	constant	
days_left	decimal	
frsn	decimal	
rsn	decimal	
rar	decimal	
status	constant	
function_support	constant	
branch_aware	constant	

If the command executes successfully and you specified `DETAIL` as the *list_options* parameter value, the following parameters are returned:

node_name
Fully qualified name of the node.

node_type
Type of the node. Possible values are:

NETWORK_NODE
Network node (NN)

END_NODE
End node (EN)

VRN Virtual routing node (VRN)

days_left
Number of days before this node entry will be deleted from the topology database. For the local node entry, this value is set to `0` (zero), indicating that this entry is never deleted.

frsn Flow reduction sequence number. Indicates the last time that this resource was updates at the local node.

rsn Resource sequence number. This number is assigned by the network node that owns this resource.

rar The route additional resistance of the node, in the range `0–255`.

status Specifies the status of the node. This parameter can be set to `UNCONGESTED`, to any one of the other values listed, or to two or more of the other values combined with a + character. Possible values are:

UNCONGESTED
The number of ISR sessions is below the *isr_sessions_upper_threshold* value in the node's configuration.

CONGESTED
The number of ISR sessions exceeds the *isr_sessions_upper_threshold* value.

IRR_DEPLETED
The number of ISR sessions has reached the maximum specified for the node.

ERR_DEPLETED
The number of end-point sessions has reached the maximum specified for the node.

QUIESCING
The node is in the process of stopping as a result of a **stop_node** command with a stop type of `QUIESCE` or `QUIESCE_ISR`.

function_support
Specifies which functions are supported by the node. Possible values are one or more of the following:

PERIPHERAL_BORDER_NODE
Peripheral border node function is supported.

EXTENDED_BORDER_NODE
Return border node function is supported.

CDS Central directory server function is supported.

GATEWAY
Gateway node function is supported.

INTERCHANGE_NODE
Interchange node function is supported.

ISR Intermediate session routing function is supported.

HPR Node supports the base functions of High Performance Routing (HPR).

RTP_TOWER
Node supports the Rapid Transport Protocol tower of HPR.

CONTROL_OVER_RTP_TOWER
Node supports HPR control flows over the Rapid Transport Protocol tower.

branch_aware
Specifies whether the node supports branch awareness, APPN Option Set 1120.

NO The node does not support option set 1120.

YES The node supports option set 1120.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_NODE
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *node_name* parameter value was not valid.

State Check

No specific state errors, other than those listed in Appendix A, “Common Return Codes from snaadmin Commands,” on page 589, occur for this command.

Function Not Supported

If the command does not execute because the node’s configuration does not support it, Communications Server for Linux returns the following parameters:

primary_rc

FUNCTION_NOT_SUPPORTED
The local node is an end node or LEN node. This command is valid only for a network node.

secondary_rc
(This parameter is not used.)

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_nn_topology_stats

The **query_nn_topology_stats** command returns statistical information about the topology database. The command can be used only if the Communications Server for Linux node is a network node and is not valid if it is an end node or LEN node.

This command must be issued to a running node.

Supplied Parameters

```
[query_nn_topology_stats]
```

No parameters are supplied for this command.

Returned Parameters

Parameter name	Type
max_nodes	decimal
cur_num_nodes	decimal
node_in_tdus	decimal
node_out_tdus	decimal
node_low_rsns	decimal
node_equal_rsns	decimal
node_good_high_rsns	decimal
node_bad_high_rsns	decimal
node_state_updates	decimal
node_errors	decimal
node_timer_updates	decimal
node_purges	decimal
tg_low_rsns	decimal
tg_equal_rsns	decimal
tg_good_high_rsns	decimal
tg_bad_high_rsns	decimal
tg_state_updates	decimal
tg_errors	decimal
tg_timer_updates	decimal
tg_purges	decimal
total_route_calcs	decimal
total_route_rejs	decimal
total_tree_cache_hits	decimal
total_tree_cache_misses	decimal
total_tdu_wars	decimal

If the command executes successfully, the following parameters are returned:

max_nodes
: Maximum number of node records in the Topology Database that were specified in the node definition. A value of `0` (zero) indicates no limit.

cur_num_nodes
: Current number of nodes in this node's topology database. If this value exceeds the maximum number of nodes allowed, an alert is issued.

node_in_tdus
: Total number of topology database updates (TDUs) received by this node.

node_out_tdus
Total number of TDUs built by this node to be sent to all adjacent network nodes since the last initialization.

node_low_rsns
Total number of topology node updates received by this node with a resource sequence number (RSN) less than the current RSN. Both even and odd RSNs are included in this count. These TDUs are not errors but occur when TDUs are broadcast to all adjacent network nodes. This node's topology database is not updated, but this node sends a TDU with its higher RSN to the adjacent node that sent the low RSN.

node_equal_rsns
Total number of topology node updates received by this node with RSN equal to the current RSN. Both even and odd RSNs are included in this count. These TDUs are not errors, but result when TDUs are broadcast to all adjacent network nodes. This node's topology database is not updated.

node_good_high_rsns
Total number of topology node updates received by this node with RSN greater than the current RSN. The node updates its topology and broadcasts a TDU to all adjacent network nodes. The node is not required to send a TDU to the sender of this update because that node already has the update.

node_bad_high_rsns
Total number of topology node updates received by this node with an odd RSN greater than the current RSN. These updates represent a topology inconsistency detected by one of the APPN network nodes. The node updates its topology and broadcasts the TDU to all adjacent network nodes.

node_state_updates
Total number of topology node updates built as a result of internally detected node state changes that affect APPN topology and routing. Node updates are sent via TDUs to all adjacent network nodes.

node_errors
Total number of topology node update inconsistencies detected by this node. A topology database update inconsistency occurs when this node attempts to update its topology database and detects a data inconsistency. This node creates a TDU with the current RSN increased to the next odd number and broadcasts it to all adjacent network nodes.

node_timer_updates
Total number of topology node updates built for this node's resource due to timer updates. Node updates are sent via TDUs to all adjacent network nodes. These updates ensure that other network nodes do not delete this node's resource from their topology database.

node_purges
Total number of topology node records purged from this node's topology database. A purge occurs when a node record has not been updated in a specified amount of time. The owning node is responsible for broadcasting updates for its resource that it wants kept in the network topology.

tg_low_rsns
Total number of topology TG updates received by this node with RSN less than the current RSN. Both even and odd RSNs are included in this count. These TDUs are not errors but occur when TDUs are broadcast to all

adjacent network nodes. This node's topology database is not updated, but this node will send a TDU with its higher RSN to the adjacent node that sent this low RSN.

tg_equal_rsns
Total number of topology TG updates received by this node with RSN equal to the current RSN. Both even and odd RSNs are included in this count. These TDUs are not errors but occur when TDUs are broadcast to all adjacent network nodes. This node's topology database is not updated.

tg_good_high_rsns
Total number of topology TG updates received by this node with RSN greater than the current RSN. The node updates its topology and broadcasts a TDU to all adjacent network nodes.

tg_bad_high_rsns
Total number of topology TG updates received by this node with an odd RSN greater than the current RSN. These updates represent a topology inconsistency detected by one of the APPN network nodes. The node updates its topology and broadcasts the TDU to all adjacent network nodes.

tg_state_updates
Total number of topology TG updates built as a result of internally detected node state changes that affect APPN topology and routing. TG updates are sent via TDUs to all adjacent network nodes.

tg_errors
Total number of topology TG update inconsistencies detected by this node. A TG update inconsistency occurs when this node attempts to update its topology database and detects a data inconsistency. This node creates a TDU with the current RSN increased to the next odd number and broadcasts it to all adjacent network nodes.

tg_timer_updates
Total number of topology TG updates built for this node's resource due to timer updates. TG updates are sent via TDUs to all adjacent network nodes. These updates ensure that other network nodes do not delete this node's resource from their topology database.

tg_purges
Total number of topology TG records purged from this node's topology database. A purge occurs when a TG record has not been updated in a specified amount of time. The owning node is responsible for broadcasting updates for its resource that it wants kept in the network topology.

total_route_calcs
Number of routes calculated for all class of services since the last initialization.

total_route_rejs
Number of route requests for all class of services that could not be calculated since the last initialization.

total_tree_cache_hits
Number of route computations that were satisfied by a cached routing tree. This number may be greater than the total number of computed routes because each route may require the inspection of several trees.

total_tree_cache_misses
Number of route computations that were not satisfied by a cached routing tree, so that a new routing tree had to be built.

total_tdu_wars
Number of TDU wars the local node has detected and prevented.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Function Not Supported

If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameter:

primary_rc

FUNCTION_NOT_SUPPORTED
The local node is an end node or LEN node. This command is valid only for a network node.

secondary_rc
(This parameter is not used.)

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_nn_topology_tg

Each network node (NN) maintains a network topology database that holds information about all the network nodes, VRNs, and network node to network node TGs in the network. The **query_nn_topology_tg** command returns information about the TG entries in this database. This command can be used to obtain summary or detailed information about a specific TG or about multiple TGs, depending on the options used.

This command must be issued to a running node. It can be used only if the Communications Server for Linux node is a network node and is not valid if it is an end node or LEN node.

Supplied Parameters

Parameter name	Type	Length	Default
[query_nn_topology_tg]			
num_entries	decimal		1
list_options	constant		SUMMARY + LIST_INCLUSIVE
owner	character	17	(null string)
owner_type	constant		LEARN_NODE
dest	character	17	(null string)

```
dest_type               constant                LEARN_NODE
tg_num                  decimal                 0
frsn                    decimal                 0
```

Note: If the *frsn* parameter is set to a nonzero value, then only node entries with FRSNs equal to or greater than the one specified will be returned. If the *frsn* parameter is set to 0 (zero), then all node entries are returned.

Supplied parameters are:

num_entries
Maximum number of TGs for which data should be returned. You can specify 1 to return data for a specific TG, a number greater than 1 to return data for multiple TGs, or 0 (zero) to return data for all TGs.

list_options
The level of information required for each entry and the position in the list of TGs from which Communications Server for Linux begins to return data. The list is ordered by *owner*, *owner_type* (in the order NETWORK_NODE, VRN), *dest*, *dest_type* (in the order NETWORK_NODE, VRN), *tg_num* (numerically), and finally *frsn* (numerically).

The combination of the *owner*, *owner_type*, *dest*, *dest_type*, *tg_num*, and *frsn* parameters specified is used as an index into the list of TGs if the *list_options* parameter is set to LIST_INCLUSIVE or LIST_FROM_NEXT.

Specify the level of information required with one of the following values:

SUMMARY
Summary information only

DETAIL Detailed information

Use a + character to combine this value with one of the following values:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the combination of *owner*, *owner_type*, *dest*, *dest_type*, *tg_num*, and *frsn*

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the combination of *owner*, *owner_type*, *dest*, *dest_type*, *tg_num*, and *frsn*

owner Name of the node that owns the TG. This value is ignored if *list_options* is set to FIRST_IN_LIST. The name is type-A character string, consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character node name.

owner_type
Type of the node that owns the TG. Possible values are:

NETWORK_NODE
Network node (NN)

VRN Virtual routing node (VRN)

LEARN_NODE
Node type is unknown

dest Name of the destination node for the TG. This value is ignored if *list_options* is set to FIRST_IN_LIST. The name is a type-A character string,

consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character destination node name.

dest_type
Type of the destination node for the TG. Possible values are:

NETWORK_NODE
Network node (NN)

VRN Virtual routing node (VRN)

LEARN_NODE
Node type is unknown

tg_num
Number associated with the TG.

frsn Flow reduction sequence number. Specify `0` (zero) to return information about all TGs or a nonzero value to return information about TGs with a FRSN greater than or equal to this value.

This parameter can be used to ensure that consistent information is obtained when you need to issue several commands to obtain all required information. Take the following steps:

To Obtain Consistent Information Using the frsn Parameter

1. Issue **query_node** to get the node's current FRSN.
2. Issue as many **query_nn_topology_node** commands as necessary to get all the database entries, with the *frsn* parameter set to `0` (zero).
3. Issue **query_node** again and compare the new FRSN with the one returned in step 1.
4. If the two FRSNs are different, the database has changed. Add 1 to the FRSN obtained in step 1, and issue additional **query_nn_topology_node** commands with the *frsn* parameter set to this new value. These commands return only the entries that have changed.

Returned Parameters: Summary Information

Parameter name	Type	Length
owner	character	17
owner_type	constant	
dest	character	17
dest_type	constant	
tg_num	decimal	
frsn	decimal	

If the command executes successfully and you specified `SUMMARY` as the *list_options* parameter value, the following parameters are returned:

owner Name of the node that owns the TG.

owner_type
Type of the node that owns the TG. Possible values are:

NETWORK_NODE
Network node (NN)

END_NODE
End node (EN)

VRN Virtual routing node (VRN)

dest Name of the destination node for the TG.

dest_type
Type of the destination node for the TG. Possible values are:

NETWORK_NODE
Network node (NN)

END_NODE
End node (EN)

VRN Virtual routing node (VRN)

tg_num
Number associated with the TG.

frsn Flow reduction sequence number indicating the last time that this resource was updated at the local node.

Returned Parameters: Detailed Information

Parameter name	Type	Length
owner	character	17
owner_type	constant	
dest	character	17
dest_type	constant	
tg_num	decimal	
frsn	decimal	
days_left	decimal	
dlc_data	hex array	32
rsn	decimal	
status	constant	
effect_cap	hex number	
connect_cost	decimal	
byte_cost	decimal	
security	constant	
prop_delay	constant	
user_def_parm_1	decimal	128
user_def_parm_2	decimal	128
user_def_parm_3	decimal	128
subarea_number	hex array	8
tg_type	constant	
intersubnet_tg	constant	
cp_cp_session_active	constant	
branch_tg	constant	
multilink_tg	constant	

If the command executes successfully and you specified `DETAIL` as the *list_options* parameter value, the following parameters are returned:

owner Name of the node that owns the TG.

owner_type
Type of the node that owns the TG. Possible values are:

NETWORK_NODE
Network node (NN)

END_NODE
End node (EN)

VRN Virtual routing node (VRN)

dest Name of the destination node for the TG.

dest_type
Type of the destination node for the TG. Possible values are:

NETWORK_NODE
Network node (NN)

END_NODE
End node (EN)

VRN Virtual routing node (VRN)

tg_num
Number associated with the TG.

frsn Flow reduction sequence number indicating the last time that this resource was updated at the local node.

days_left
Number of days before this TG entry will be deleted from the Topology Database.

dlc_data
If *dest_type* or *owner_type* is `VRN`, this parameter specifies the DLC address of the connection to the VRN. The number of bytes in the address depends on the DLC type. This parameter is not used otherwise.

For Token Ring or Ethernet, the address is in two parts: a 6-byte MAC address and a 1-byte local SAP address. The bit ordering of the MAC address may not be in the expected format. For information about converting between the two address formats, see "Bit Ordering in MAC Addresses" on page 225.

rsn Resource sequence number assigned by the network node that owns this resource.

status Specifies the status of the TG. Possible values are:

NONE Transmission group link is not established.

TG_OPERATIVE
Transmission group link is operative.

TG_CP_CP_SESSIONS
Transmission group link is operative and carrying CP-CP sessions.

TG_QUIESCING
Transmission group link is shutting down.

TG_HPR Transmission group supports High Performance Routing (HPR) protocols.

TG_RTP Transmission group supports Rapid Transport Protocol (RTP).

effect_cap **through** *user_def_parm_3*
Default TG characteristics used for implicit link stations using this port and as the default TG characteristics for defined link stations that do not have explicitly defined TG characteristics. The TG characteristics parameters are ignored if the LS is to a downstream PU.

For more information about these parameters, see "define_tr_ls, define_ethernet_ls" on page 209.

subarea_number
If the owner of the destination of the TG is subarea capable, this parameter contains the subarea number of the type-4 or type-5 node that owns the link station associated with the TG on the subarea capable node. Otherwise, this parameter is set to all binary zeros.

tg_type
Type of the TG. Possible values are:

APPN_OR_BOUNDARY_TG
APPN TG or boundary function based TG.

INTERCHANGE_TG
Interchange TG.

VIRTUAL_ROUTE_BASED_TG
Virtual route based TG.

UNKNOWN
The TG type is unknown.

intersubnet_tg
Specifies whether the TG is an intersubnetwork TG. Possible values are:

YES The TG is an intersubnetwork TG.

NO The TG is not an intersubnetwork TG.

cp_cp_session_active
Specifies whether the owning node's contention winner CP-CP session is active. Possible values are:

YES The CP-CP session is active.

NO The CP-CP session is not active.

UNKNOWN
The CP-CP session status is unknown.

branch_tg
Specifies whether the TG is a branch TG. Possible values are:

YES The TG is a branch TG.

NO The TG is not a branch TG.

UNKNOWN
The TG type is unknown.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_TG
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *tg_num* parameter value was not valid.

INVALID_ORIGIN_NODE
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *owner* parameter value was not valid.

State Check
No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Function Not Supported
If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameter:

primary_rc

FUNCTION_NOT_SUPPORTED
The local node is an end node or LEN node. This command is valid only for a network node.

secondary_rc
(This parameter is not used.)

Other Conditions
Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

query_node

The **query_node** command returns information about the definition of a Communications Server for Linux node and also on its status if it is active. This command returns information only about a single node. To obtain a list of nodes in the Communications Server for Linux domain, use the command **query_node_all**; you can then use **query_node** for an individual node in this list to obtain more detailed information.

Supplied Parameters
[query_node]

No parameters are supplied for this command.

Returned Parameters

Parameter name	Type	Length
description	character	31
node_type	constant	
fqcp_name	character	17
cp_alias	character	8
mode_to_cos_map_supp	constant	
mds_supported	constant	
node_id	character	4
max_locates	decimal	
dir_cache_size	decimal	
max_dir_entries	decimal	
locate_timeout	decimal	
reg_with_nn	constant	
reg_with_cds	constant	
mds_send_alert_q_size	decimal	
cos_cache_size	decimal	
tree_cache_size	decimal	

```
 tree_cache_use_limit               decimal
 max_tdm_nodes                      decimal
 max_tdm_tgs                        decimal
 max_isr_sessions                   decimal
 isr_sessions_upper_threshold       decimal
 isr_sessions_lower_threshold       decimal
 isr_max_ru_size                    decimal
 isr_rcv_pac_window                 decimal
 store_endpt_rscvs                  constant
 store_isr_rscvs                    constant
 store_dlur_rscvs                   constant
 dlur_support                       constant
 pu_conc_support                    constant
 nn_rar                             decimal
 hpr_support                        constant
 ptf_flags                          constant
 max_ls_exception_events            decimal
 max_compress_level                 constant
 clear_initial_topology             constant
 up_time                            decimal
 nn_functions_supported             constant
 en_functions_supported             constant
 nn_status                          constant
 nn_frsn                            decimal
 nn_rsn                             decimal
 def_ls_good_xids                   decimal
 def_ls_bad_xids                    decimal
 dyn_ls_good_xids                   decimal
 dyn_ls_bad_xids                    decimal
 dlur_release_level                 decimal
 fq_nn_server_name                  character          17
 current_isr_sessions               decimal
nns_dlus_served_lu_reg_supp         constant
nn_functions2                       constant
branch_ntwk_arch_version            decimal
```

If the command executes successfully, the following parameters are returned:

description
A text string describing the node, as specified in the definition of the node.

node_type
Type of node. Possible values are:

LEN_NODE
Low entry networking (LEN) node

END_NODE
APPN end node

NETWORK_NODE
APPN network node

BRANCH_NETWORK_NODE
APPN branch network node

fqcp_name
Fully qualified CP name of the node.

cp_alias
Locally used CP alias.

mode_to_cos_map_supp
Specifies whether the node provides mode-to-COS mapping. This

parameter is ignored for a network node; mode-to-COS mapping is always supported. For a LEN node, mode-to-COS mapping is not supported. Possible values are:

YES Mode-to-COS mapping is supported. A mode defined for this node must include its associated COS name, which specifies either an SNA-defined COS or one defined using **define_cos**.

NO Mode-to-COS mapping is not supported. Default COS names will be used.

mds_supported
Specifies whether Management Services (MS) supports Multiple Domain Support (MDS) and Management Services Capabilities. Possible values are:

YES MDS is supported.

NO MDS is not supported.

node_id
Node identifier used in XID exchange. This ID is a 4-byte hexadecimal string, consisting of a block number (three hexadecimal digits) and a node number (five hexadecimal digits).

max_locates
Maximum number of locate requests (requests for which a response has not yet been received) that the node can process simultaneously. When the number of outstanding locate requests reaches this limit, any further locate requests are rejected.

dir_cache_size
Network node only: Size of the directory cache. The minimum size is 3.

max_dir_entries
Maximum number of directory entries. The value `0` (zero) indicates no limit.

locate_timeout
Specifies the time in seconds before a network search will time out. The value `0` (zero) indicates no time out.

reg_with_nn
End node only: Specifies whether the node's resources are registered with the network node server when the node is started. Possible values are:

YES Resources are registered with the network node. The end node's network node server will only forward directed locates to the network node.

NO Resources are not registered. The network node server will forward all broadcast searches to the end node.

reg_with_cds
End node: Specifies whether the network node server is allowed to register end node resources with a central directory server. This parameter is ignored if *reg_with_nn* is set to `NO`.

Network node: Specifies whether local or domain resources can be optionally registered with central directory server.

Possible values are:

YES Resources are registered with the CDS.

NO Resources are not registered.

mds_send_alert_q_size
Size of the MDS send alert queue. If the number of queued alerts reaches this limit, Communications Server for Linux deletes the oldest alert on the queue. The minimum size is 2.

cos_cache_size
Size of the COS Database weights cache.

tree_cache_size
Network node: Size of the Topology Database routing tree cache. The minimum is 8. For an end node or LEN node, this parameter is reserved.

tree_cache_use_limit
Network node: Maximum number of uses of a cached tree. When this number is exceeded, the tree is discarded and recomputed. This enables the node to balance sessions among equal weight routes. A low value provides better load balancing at the expense of increased activation latency. The minimum number of uses is 1. For an end node or LEN node, this parameter is reserved.

max_tdm_nodes
Network node: Maximum number of nodes that can be stored in Topology Database. A value of `0` (zero) indicates an unlimited number of nodes. For an end node or LEN node, this parameter is reserved.

max_tdm_tgs
Network node: Maximum number of TGs that can be stored in Topology Database. A value of `0` (zero) indicates an unlimited number of nodes. For an end node or LEN node, this parameter is reserved.

max_isr_sessions
Network node: Maximum number of ISR sessions the node can participate in at once. This parameter is reserved for an end node or LEN node.

isr_sessions_upper_threshold **and** *isr_sessions_lower_threshold*
Network node: These thresholds control the node's congestion status, which is reported to other nodes in the network for use in route calculations. The node state changes from uncongested to congested if the number of ISR sessions exceeds the upper threshold. The node state changes back to uncongested when the number of ISR sessions dips below the lower threshold. For an end node or LEN node, these parameters are reserved.

isr_max_ru_size
Network node: Maximum RU size supported for intermediate sessions. For an end node or LEN node, this parameter is reserved.

isr_rcv_pac_window
Network node: Suggested receive pacing window size for intermediate sessions, in the range 1–63. This value is only used on the secondary hop of intermediate sessions if the adjacent node does not support adaptive pacing. For an end node or LEN node, this parameter is reserved.

store_endpt_rscvs
Specifies whether RSCVs are stored for diagnostic purposes. If this parameter is set to `YES`, an RSCV is returned on the **query_session** command; an RSCV is stored for each endpoint session. This extra storage can be up to 256 bytes per session. Possible values are:

YES RSCVs are stored.

NO RSCVs are not stored.

store_isr_rscvs
Network node: Specifies whether RSCVs are stored for diagnostic purposes. If this parameter is set to YES, an RSCV is returned on the **query_isr_session** command; an RSCV is stored for each ISR session. This extra storage can be up to 256 bytes per session. Possible values are:

YES RSCVs are stored.

NO RSCVs are not stored.

store_dlur_rscvs
Specifies whether RSCVs are stored for diagnostic purposes. If this parameter is set to YES, an RSCV is returned on the **query_dlur_lu** command; an RSCV is stored for each PLU-SLU session using DLUR. This extra storage can be up to 256 bytes per session. Possible values are:

YES RSCVs are stored.

NO RSCVs are not stored.

dlur_support
Specifies whether DLUR is supported. For a LEN node, this parameter is reserved. Possible values are:

YES DLUR is supported.

LIMITED_MULTI_SUBNET
End Node: DLUR is supported, but will not be used to connect to a DLUS in another subnet.

This value is not supported for a Network Node.

NO DLUR is not supported.

pu_conc_support
Specifies whether SNA gateway is supported. Possible values are:

YES SNA gateway is supported.

NO SNA gateway is not supported.

nn_rar The network node's route additional resistance. Values must be in the range 0–255.

hpr_support
Specifies the level of HPR (High Performance Routing) support provided by the node. Possible values are:

NONE No support for HPR.

BASE This node can perform automatic network routing (ANR) but cannot act as an RTP (Rapid Transport Protocol) end point for HPR sessions.

RTP This node can perform automatic network routing (ANR) and can act as an RTP (Rapid Transport Protocol) end point for HPR sessions.

CONTROL_FLOWS
This node can perform all HPR functions including control flows.

ptf_flags
Options for configuring and controlling program temporary fix (ptf) operation. This parameter is set to NONE or to one or more of the following values which can be combined with a + character.

Possible values are:

ERP Communications Server for Linux normally processes an ACTPU(ERP) as an ERP; this resets the PU-SSCP session but does not implicitly deactivate the subservient LU-SSCP and PLU-SLU sessions. SNA implementations may legally process ACTPU(ERP) as if it were ACTPU(cold), implicitly deactivating the subservient LU-SSCP and PLU-SLU sessions.

BIS Communications Server for Linux normally uses the BIS protocol prior to deactivating a limited resource LU 6.2 session.

OVERRIDE_REQDISCONT
Communications Server for Linux normally uses REQDISCONT to deactivate limited resource host links that are no longer required by session traffic.

If OVERRIDE_REQDISCONT is specified, it is combined with one or both of the following values to alter the type of the REQDISCONT message:

- IMMEDIATE_DISCONTACT: Communications Server for Linux uses type "immediate" on REQDISCONT; if this value is not specified, Communications Server for Linux uses type "normal."
- IMMEDIATE_RECONTACT: Communications Server for Linux uses type "immediate recontact" on REQDISCONT; if this value is not specified, Communications Server for Linux uses type "no immediate recontact."

SUPPRESS_REQDISCONT
Limited resource host links are deactivated without sending REQDISCONT.

ALLOW_BB_RQE
Communications Server for Linux normally rejects, with sense code 2003, any begin bracket (BB) exception (RQE) request from a host unless the host follows the SNA protocol that the request must also indicate change direction (CD) . Setting this flag enables Communications Server for Linux to continue sessions with hosts that do not follow this protocol.

EXTERNAL_APINGD
Communications Server for Linux normally includes a partner program for the APING connectivity tester. Setting this value disables the APING Daemon within the node. Requests by an APING program arriving at the node will not be processed automatically.

SET_SEARCH_STATUS
When Communications Server for Linux is running as an End Nodeor as a Branch Network Node, it may choose whether or not to invite network searches from its Network Node Server (NNS). Requesting network searches slows broadcast search processing for the network as a whole, so is undesirable. However, if the local node cannot register all its resources (LUs) with its NNS, requesting searches is the only way to make these resources visible to the network.

Normally, Communications Server for Linux determines whether all LUs can be registered, then intelligently requests network searches from its NNS. If this node makes LUs accessible to the

network in an unusual manner (for example, if it is acting as a gateway for other nodes), the value SET_SEARCH_STATUS overrides the standard operation.

LIMIT_TP_SECURITY
Security checking for received Attaches. If a local invokable TP is defined not to require conversation security, or is not defined and therefore defaults to not requiring conversation security, the invoking TP need not send a user ID and password to access it. If the invoking TP supplies these parameters and they are included in the Attach message that Communications Server for Linux receives, Communications Server for Linux normally checks the parameters (and rejects the Attach if they are not valid) even though the invokable TP does not require conversation security. This value disables the checking, so that Communications Server for Linux does not check security parameters on a received Attach if the invokable TP does not require it.

FORCE_STANDARD_ARB
Communications Server for Linux normally advertises support on RTP connections for both the standard ARB algorithm and the ARB responsive mode algorithm. If this value is set, Communications Server for Linux will only advertise support for the standard ARB algorithm.

DLUR_UNBIND_ON_DACTLU
Communications Server for Linux does not normally end the PLU-SLU session when it receives a DACTLU from the host for a session using DLUR. If this value is set, Communications Server for Linux will end the PLU-SLU session when it receives a DACTLU from the host for a session using DLUR.

NO_TCPIP_VECTOR
Communications Server for Linux normally includes the TCP/IP Information Control Vector (0x64) in a NOTIFY request to the host for a TN3270 or LUA session. This vector contains information that can be displayed on the host console or used by the host (for example in billing): the TCP/IP address and port number used by the client, and the IP name corresponding to the client address.

If the client address is an IPv6 address but the host is running a back-level version of VTAM that cannot interpret IPv6 addresses, the client address may be displayed incorrectly on the host console.

In some cases, for example if the host is running an older version of VTAM that does not support this vector, you may need to override this behavior so that the vector is not sent. This flag suppresses sending the vector to the host.

NO_TCPIP_NAME
Communications Server for Linux TN Server normally performs a Domain Name Server (DNS) lookup to determine the client IP name for inclusion in the TCP/IP Information Control Vector (0x64) as described above. You may want to avoid this DNS lookup if the DNS environment is slow, or if you know that the clients are not included in the DNS data (for example if they are DHCP clients without DDNS). This flag suppresses the DNS lookup; Communications Server for Linux TN Server will send the CV64 control vector with the client IP address but no IP name.

This value applies only to TN3270; no DNS lookup is required for LUA clients.

max_ls_exception_events
The maximum number of LS exception events recorded by the node.

max_compress_level
The maximum compression supported by the node for LU session data. This parameter is always set to `LZ10`.

clear_initial_topology
Indicates whether starting the node clears any topology data that was stored when it was last active. Possible values are:

YES Clear the stored topology data.

NO Keep any topology data that was stored when the node was last active, so that it can be re-used.

up_time
Time, in hundredths of a second, since the node was started. If this parameter is `0` (zero), this indicates that the node is inactive.

nn_functions_supported
Specifies the network node functions that are supported. This parameter can be one or more of the following values, which can be combined with a + character:

RCV_REG_CHAR
Node supports receiving registered characteristics.

GATEWAY
Node is a gateway node.

CDS Node supports central directory server (CDS) function.

TREE_CACHING
Node supports route tree cache.

TREE_UPDATES
Node supports incremental tree updates. If incremental tree updates are supported, tree caching must also be supported.

ISR Node supports ISR.

This parameter is reserved for an end node or LEN node.

en_functions_supported
Specifies the end node functions that are supported. This parameter can be one or more of the following values, which can be combined with a + character:

SEGMENT_GENERATION
Node supports segment generation.

MODE_TO_COS_MAP
Node supports mode name-to-COS name mapping.

LOCATE_CDINIT
Node supports generation of locates and cross-domain initiate GDS variables for locating remote LUs.

REG_WITH_NN
Node will register its LUs with the adjacent serving network node.

This parameter is reserved for a network node or LEN node.

nn_status
Specifies the status of the network node. This parameter is reserved if the node is not a network node.

This parameter can be set to UNCONGESTED or to one or more of the following values which can be combined with a + character:

UNCONGESTED
The number of ISR sessions is below the *isr_sessions_upper_threshold* value in the node's configuration.

CONGESTED
The number of ISR sessions exceeds the threshold value.

IRR_DEPLETED
The number of ISR sessions has reached the maximum number specified for the node.

ERR_DEPLETED
The number of end-point sessions has reached the maximum number specified.

QUIESCING
A **term_node** command has been issued with a stop type of QUIESCE or QUIESCE_ISR.

nn_frsn
The current flow reduction sequence number of the network node.

This parameter is reserved if the node is not a network node.

nn_rsn The resource sequence number of the network node.

This parameter is reserved if the node is not a network node.

def_ls_good_xids
Total number of successful XID exchanges that have occurred on all defined link stations since the node was last started.

def_ls_bad_xids
Total number of unsuccessful XID exchanges that have occurred on all defined link stations since the node was last started.

dyn_ls_good_xids
Total number of successful XID exchanges that have occurred on all dynamic link stations since the node was last started.

dyn_ls_bad_xids
Total number of unsuccessful XID exchanges that have occurred on all dynamic link stations since the node was last started.

dlur_release_level
Release level of the DLUR architecture supported by the node. This parameter is set to 1 (the only release level of DLUR currently defined); future versions may incorporate later release levels of the DLUR architecture and therefore may return different values.

fq_nn_server_name
End node only. Name of the network node server for the node.

current_isr_sessions
Number of ISR sessions routed through this node.

nns_dlus_served_lu_reg_supp
This parameter applies only if the local node is an end node or a Branch Network Node; it is reserved otherwise.

Specifies whether the network node server supports DLUS-served LU registration. Possible values are:

YES The network node server supports registration of DLUS-served LUs.

NO The network node server does not support registration of DLUS-served LUs.

UNKNOWN
The node does not have a network node server.

nns_en_reg_diff_owning_cp
This parameter applies only if the local node is a Branch Network Node; it is reserved otherwise.

Specifies whether the network node server supports option set 1123 - End Node Resource Registration With Different Owning CP Name NNS(BrNN) Support.

YES The network node server supports option set 1123.

NO The network node server does not support option set 1123.

UNKNOWN
The node does not have a network node server.

nn_functions_2
This parameter applies only if the local node is a Network Node; it is reserved otherwise.

If the node supports branch awareness, APPN Option Set 1120, this parameter is set to the following value:

BRANCH_AWARENESS
The node supports option set 1120.

branch_ntwk_arch_version
This parameter applies only if the local node is a Network Node or a Branch Network Node; it is reserved otherwise.

Specifies the version of the Branch Network Architecture supported. This is set to `1`, or `0` (zero) if the node does not support the Branch Network Architecture.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists additional combinations of primary and secondary return codes that are common to all commands.

query_node_all

The **query_node_all** command returns information about nodes in the Communications Server for Linux domain. The command returns only a list of node names and does not provide detailed information about the node's configuration. You can use **query_node** for a particular node name to obtain detailed information about that node.

This command must be issued without using the **-n** option of the **snaadmin** program.

Supplied Parameters

Parameter name	Type	Length	Default
[query_node_all]			
num_entries	decimal		1
list_options	constant		LIST_INCLUSIVE
node_name	character	128	(null string)

Supplied parameters are:

num_entries
: Maximum number of nodes for which data should be returned. You can specify 1 to return data for a specific node, a number greater than 1 to return data for multiple nodes, or 0 (zero) to return data for all nodes.

list_options
: The position in the list of nodes from which Communications Server for Linux begins to return data. The list is not ordered by node name; however, the order remains the same for subsequent **query_node_all** commands.

  Possible values are:

  FIRST_IN_LIST
  : Start at the first entry in the list of nodes

  LIST_INCLUSIVE
  : Start at the entry specified by the *node_name* parameter

  LIST_FROM_NEXT
  : Start at the entry immediately following the entry specified by the *node_name* parameter

node_name
: Name of the node to be used as an index into the list. This parameter is ignored if *list_options* is set to FIRST_IN_LIST.

  If the computer name includes a . (period) character, Communications Server for Linux assumes that it is a fully-qualified name; otherwise it performs a DNS lookup to determine the computer name.

Returned Parameters

Parameter name	Type	Length
node_name	character	128
config_role	constant	

If the command executes successfully, Communications Server for Linux returns the following parameters:

node_name
The name of the Communications Server for Linux node.

config_role
The configuration file role of the server where the node is running. For more information about configuration file roles, refer to the *Communications Server for Linux Administration Guide*. Possible values are:

MASTER The server holds the master configuration file.

BACKUP The server holds a backup configuration file.

NONE The server does not share its copy of the configuration file.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_NODE_NAME
The *list_options* parameter was set to LIST_INCLUSIVE or LIST_FROM_NEXT to list all entries starting from the supplied node name, but the *node_name* parameter was not specified or was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_node_limits

The **query_node_limits** command returns information about the functions that your Communications Server for Linux license allows you to use on a particular node, and about your usage of these functions. These are divided into the following categories:

- Node options, which specify the Communications Server for Linux features that you can use
- Node resource usage, which specifies the current and peak usage of Communications Server for Linux resources.

You can use the information returned by this command to check whether your usage of Communications Server for Linux resources is within the limits permitted by your license. For more information about licensing requirements, see *Communications Server for Linux Quick Beginnings*.

The information returned by this command is also written to the usage log file at intervals. For more information about this file, see *Communications Server for Linux Diagnostics Guide*.

Supplied Parameters

```
[query_node_limits]
```

No parameters are supplied for this command.

Returned Parameters

Parameter name	Type
current_lu62_tps	decimal
current_lua_tps	decimal
current_link_stations	decimal
current_tn3270_connections	decimal
current_tn_redirector_connections	decimal
current_data_sessions	decimal
peak_lu62_tps	decimal
peak_lua_tps	decimal
peak_link_stations	decimal
peak_tn3270_connections	decimal
peak_tn_redirector_connections	decimal
peak_data_sessions	decimal
branch_network_node	constant
network_node	constant
end_node	constant
len_node	constant
dlur_support	constant
pu_conc_support	constant
tn_server_support	constant
hpr_support	constant
back_level_client	constant
ssl_support	constant

If the command executes successfully, the following parameters are returned:

current_lu62_tps
The number of APPC and CPI-C applications currently active on this node.

current_lua_tps
The number of LUA applications currently active on this node.

current_link_stations
The number of link stations currently active on this node.

current_tn3270_connections
The number of connections from TN3270 clients currently active on this node.

current_tn_redirector_connections
The number of connections from TN Redirector clients currently active on this node.

current_data_sessions
The number of PLU-SLU sessions currently active on this node.

If full-duplex APPC conversations are being used, note that each full-duplex conversation requires two sessions.

peak_lu62_tps
The maximum number of APPC and CPI-C applications that have been active on this node at any time since the Linux computer was restarted.

peak_lua_tps
The maximum number of LUA applications that have been active on this node at any time since the Linux computer was restarted.

peak_link_stations
The maximum number of link stations that have been active on this node at any time since the Linux computer was restarted.

peak_tn3270_connections
The maximum number of connections from TN3270 clients that have been active on this node at any time since the Linux computer was restarted.

peak_tn_redirector_connections
The maximum number of connections from TN Redirector clients that have been active on this node at any time since the Linux computer was restarted.

peak_data_sessions
The maximum number of PLU-SLU sessions that have been active on this node at any time since the Linux computer was restarted.

If full-duplex APPC conversations are being used, note that each full-duplex conversation requires two sessions.

branch_network_node
Specifies whether your license allows you to define this node as a branch network node. Possible values are:

AP_YES Branch network node is supported.

AP_NO Branch network node is not supported.

network_node
Specifies whether your license enables you to define this node as a network node. Possible values are:

YES Your license allows you to configure this node as a network node.

NO Your license does not allow you to configure this node as a network node.

end_node
Specifies whether your license enables you to define this node as an end node. Possible values are:

YES Your license allows you to configure this node as an end node.

NO Your license does not allow you to configure this node as an end node.

len_node
Specifies whether your license enables you to define this node as a LEN node. Possible values are:

YES Your license allows you to configure this node as a LEN node.

NO Your license does not allow you to configure this node as a LEN node.

dlur_support
Specifies whether your license allows you to use Dependent LU Requester (DLUR) on this node. Possible values are:

YES Your license allows this node to provide DLUR support.

NO Your license does not allow this node to provide DLUR support.

pu_conc_support
Specifies whether your license allows you to use SNA gateway on this node. Possible values are:

YES Your license allows this node to provide SNA gateway support.

NO Your license does not allow this node to provide SNA gateway support.

tn_server_support
Specifies whether your license allows you to use TN server on this node. Possible values are:

YES Your license allows this node to provide TN server support.

NO Your license does not allow this node to provide TN server support.

hpr_support
Indicates whether HPR is supported on this node. Possible values are:

YES HPR is supported.

NO HPR is not supported.

back_level_client
This parameter is reserved.

ssl_support
Specifies whether the Secure Sockets Layer software is installed on the node (for use with TN Server). Possible values are:

YES The SSL software is installed.

NO The SSL software is not installed.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, “Common Return Codes from snaadmin Commands,” on page 589, occur for this command.

Other Conditions

Appendix A, “Common Return Codes from snaadmin Commands,” on page 589, lists additional combinations of primary and secondary return codes that are common to all commands.

query_partner_lu

The **query_partner_lu** command returns information about partner LUs that a local LU is currently using or has used. It returns information about usage of the partner LUs, not about their definition; use **query_partner_lu_definition** to obtain the definition of the partner LUs. This command can be used to obtain summary or detailed information about a specific LU or about multiple LUs, depending on the options used.

This command must be issued to a running node.

Supplied Parameters

Parameter name	Type	Length	Default
[query_partner_lu]			
num_entries	decimal		1
list_options	constant		SUMMARY + LIST_INCLUSIVE
lu_name	character	8	(null string)
lu_alias	character	8	(null string)
plu_alias	character	8	(null string)
fqplu_name	character	17	(null string)
active_sessions	constant		NO

Supplied parameters are:

num_entries
: Maximum number of LUs for which data should be returned. You can specify 1 to return data for a specific LU, a number greater than 1 to return data for multiple LUs, or `0` (zero) to return data for all LUs.

list_options
: The level of information required for each entry and the position in the list of LUs from which Communications Server for Linux begins to return data. The list is ordered by *fqplu_name*.

  Specify the level of information required with one of the following values:

  SUMMARY
  : Summary information only

  DETAIL Detailed information

  Use a + character to combine this value with one of the following values:

  FIRST_IN_LIST
  : Start at the first entry in the list of partner LUs associated with the specified local LU

  LIST_INCLUSIVE
  : Start at the entry specified by the combination of local and partner LU names

  LIST_FROM_NEXT
  : Start at the entry immediately following the entry specified by the combination of local and partner LU names

  If `FIRST_IN_LIST` is specified, you can use a + character to include the following option:

  LIST_BY_ALIAS
  : The list is returned in order of LU alias rather than LU name. This option is only valid if `FIRST_IN_LIST` is also specified. (For

LIST_FROM_NEXT or LIST_INCLUSIVE, the list is in order of LU alias or LU name, depending on which was specified as the index into the list.)

If the *list_options* parameter is set to LIST_INCLUSIVE or LIST_FROM_NEXT, the combination of the local LU (*lu_name* or *lu_alias*) and partner LU (*plu_alias* or *fqplu_name*) specified is used as an index into the list of LUs.

lu_name
LU name of the local LU. This name is a type-A character string. To indicate that the LU is identified by its LU alias instead of its LU name, do not specify this parameter, and specify the LU alias in the following parameter. To indicate the LU associated with the local CP (the default LU), do not specify either *lu_name* or *lu_alias*.

lu_alias
LU alias of the local LU. This parameter is used only if the *lu_name* parameter is not specified. To indicate the LU associated with the local CP (the default LU), do not specify either *lu_name* or *lu_alias*.

plu_alias
Partner LU alias. If *list_options* is set to FIRST_IN_LIST, this parameter is ignored; otherwise, you must specify either the LU alias or the fully qualified LU name for the partner LU. To indicate that the LU is identified by its fully qualified name instead of its alias, do not specify this parameter, and specify the LU name in the *fqplu_name* parameter.

fqplu_name
Fully qualified name of the partner LU. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character partner LU name. For the network name and partner LU name, use only letters, digits 0–9, and special characters $, #, and @.

The name can be used as the partner LU name for which information is required or as an index into the list of LUs. If *list_options* is set to FIRST_IN_LIST, this parameter is ignored; otherwise, you must specify either the LU alias or the fully qualified LU name for the partner LU. This parameter is used only if the *plu_alias* parameter is not specified; it is ignored if *plu_alias* is specified.

active_sessions
Specifies whether to return information only about partner LUs for which sessions are active, or about all partner LUs. Possible values are:

YES Return information only about partner LUs for which sessions are currently active.

NO Return information about all partner LUs for which sessions are active or have been active.

Returned Parameters: Summary Information

Parameter name	Type	Length
plu_alias	character	8
fqplu_name	character	17
description	character	31
act_sess_count	decimal	
partner_cp_name	character	17
partner_lu_located	constant	

If the command executes successfully and you specified SUMMARY as the *list_options* parameter value, the following parameters are returned:

plu_alias
Partner LU alias.

fqplu_name
A 17-byte fully qualified network name of the partner LU.

description
A text string describing the partner LU, as specified in the definition of the partner LU.

act_sess_count
Total number of active sessions between the local LU and the partner LU.

partner_cp_name
Fully qualified network name for the CP associated with the partner LU. This parameter is not used if *partner_lu_located* is set to NO.

partner_lu_located
Specifies whether the local node has located the CP where the partner LU is located. Possible values are:

YES The partner LU has been located. The *partner_cp_name* parameter contains the CP name of the partner LU.

NO The partner LU has not yet been located. The *partner_cp_name* parameter is not used.

Returned Parameters: Detailed Information

Parameter name	Type	Length
plu_alias	character	8
fqplu_name	character	17
description	character	31
act_sess_count	decimal	
partner_cp_name	character	17
partner_lu_located	constant	
plu_un_name	character	8
parallel_sess_supp	constant	
conv_security	constant	
max_mc_ll_send_size	decimal	
implicit	constant	
security_details	constant	
duplex_support	constant	
preference	constant	

If the command executes successfully and you specified DETAIL as the *list_options* parameter value, the following parameters are returned:

plu_alias
Partner LU alias.

fqplu_name
A 17-byte fully qualified network name of the partner LU.

description
A text string describing the partner LU, as specified in the definition of the partner LU.

act_sess_count
Total number of active sessions between the local LU and the partner LU.

partner_cp_name
Fully qualified network name for the CP associated with the partner LU. This parameter is not used if *partner_lu_located* is set to NO.

partner_lu_located
Specifies whether the local node has located the CP where the partner LU is located. Possible values are:

YES The partner LU has been located. The *partner_cp_name* parameter contains the CP name of the partner LU.

NO The partner LU has not yet been located. The *partner_cp_name* parameter is not used.

plu_un_name
Uninterpreted name of the partner LU.

parallel_sess_supp
Specifies whether parallel sessions are supported. Possible values are:

YES Parallel sessions are supported.

NO Parallel sessions are not supported.

conv_security
Specifies whether conversation security information supplied by a local TP can be sent to this partner LU. Possible values are:

YES Conversation security information supplied by a local TP can be sent to the partner LU.

NO Conversation security information supplied by a local TP cannot be sent to the partner LU.

UNKNOWN
No sessions are active with the partner LU.

max_mc_ll_send_size
Maximum logical record size, in bytes, that can be sent to the partner LU. This can be in the range `1`–`32,767`, or `0` (zero) to indicate no limit (in which case the maximum logical record size is 32,767 bytes). Data records that are larger than this are broken down into several LL records before being sent to the partner LU.

implicit
Specifies whether the entry was created by an implicit or explicit definition. Possible values are:

YES The entry is an implicit entry.

NO The entry is an explicit entry.

security_details
Specifies the conversation security support as negotiated on the BIND. This parameter can be set to one or more of the following values, which can be combined with a + character:

CONVERSATION_LEVEL_SECURITY
Conversation security information is accepted on requests to or from the partner LU to allocate a conversation.

ALREADY_VERIFIED
Both local and partner LU agree to accept Already Verified requests to allocate a conversation. An Already Verified request needs to carry only a user ID. It does not need to carry a password.

PERSISTENT_VERIFICATION
Persistent Verification is supported on the session between the local and partner LUs. Once the initial request (carrying a user ID and

usually a password) for a conversation has been verified, subsequent requests for a conversation need to carry only the user ID.

PASSWORD_SUBSTITUTION
The local and partner LU support Password Substitution conversation security. When a request to allocate a conversation is issued, the request carries an encrypted form of the password. If Password Substitution is not supported, the password is carried in clear text (nonencrypted) format. If the session does not support Password Substitution, an Allocate or Send_Conversation with security type set to PGM_STRONG will fail.

UNKNOWN
No sessions are active with the partner LU.

duplex_support
Returns the conversation duplex support as negotiated on the BIND. Possible values are:

HALF_DUPLEX
Only half-duplex conversations are supported.

FULL_DUPLEX
Both full-duplex and half-duplex sessions are supported. Expedited data is also supported.

UNKNOWN
The conversation duplex support is not known because no sessions are active with the partner LU.

preference
This parameter is reserved.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_LU_ALIAS
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *lu_alias* parameter value was not valid.

INVALID_LU_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *lu_name* parameter value was not valid.

INVALID_PLU_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but one of the following conditions applies:

- The *fqplu_name* parameter does not match the name of any of this local LU's partners.
- No sessions have been active since the node was last started for the specified combination of local LU and partner LU.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

query_partner_lu_definition

The **query_partner_lu_definition** command returns information about partner LUs for a local LU. This command returns information about the definition of the LUs, not about their current usage; use **query_partner_lu** to obtain the usage information. This command can be used to obtain summary or detailed information about a specific LU or about multiple LUs, depending on the options used.

Supplied Parameters

Parameter name	Type	Length	Default
[query_partner_lu_definition]			
num_entries	decimal		1
list_options	constant		SUMMARY + LIST_INCLUSIVE
plu_alias	character	8	(nullstring)
fqplu_name	character	17	(null string)

Supplied parameters are:

num_entries
Maximum number of LUs for which data should be returned. You can specify 1 to return data for a specific LU, a number greater than 1 to return data for multiple LUs, or 0 (zero) to return data for all LUs.

list_options
The level of information required for each entry and the position in the list of LUs from which Communications Server for Linux begins to return data. If *list_options* specifies FIRST_IN_LIST, the list is ordered by *plu_alias*. Otherwise, the list is ordered by *plu_alias* if it is specified, or by *fqplu_name* if it is specified.

Specify the level of information required with one of the following values:

SUMMARY
Summary information only

DETAIL Detailed information

Use a + character to combine this value with one of the following values:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the *plu_alias* or *fqplu_name* parameter

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *plu_alias* or *fqplu_name* parameter

If FIRST_IN_LIST is specified, you can use a + character to include the following option:

LIST_BY_ALIAS
The list is returned in order of LU alias rather than LU name. This option is only valid if FIRST_IN_LIST is also specified. (For LIST_FROM_NEXT or LIST_INCLUSIVE, the list is in order of LU alias or LU name, depending on which was specified as the index into the list.)

plu_alias
Partner LU alias. If *list_options* is set to FIRST_IN_LIST, this parameter is ignored; otherwise, you must specify either the LU alias or the fully qualified LU name for the partner LU. To indicate that the partner LU is defined by its fully qualified name instead of its alias, do not specify this parameter, and specify the *fqplu_name* parameter.

fqplu_name
Fully qualified name of the partner LU. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character partner LU name. For the network name and partner LU name, use only letters, digits 0–9, and special characters $, #, and @.

The name can be used as the partner LU name for which information is required or as an index into the list of LUs. If *list_options* is set to FIRST_IN_LIST, this parameter is ignored; otherwise, you must specify either the LU alias or the fully qualified LU name for the partner LU. This parameter is used only if the *plu_alias* parameter is not specified; it is ignored if *plu_alias* is specified.

Returned Parameters: Summary Information

Parameter name	Type	Length
plu_alias	character	8
fqplu_name	character	17
description	character	31

If the command executes successfully and you specified SUMMARY as the *list_options* parameter value, the following parameter is returned:

plu_alias
Partner LU alias.

fqplu_name
A 17-byte fully qualified network name of the partner LU.

description
A text string describing the partner LU, as specified in the definition of the partner LU.

Returned Parameters: Detailed Information

Parameter name	Type	Length
plu_alias	character	8
fqplu_name	character	17
description	character	31
plu_un_name	character	8
max_mc_ll_send_size	decimal	
conv_security_ver	constant	
parallel_sess_supp	constant	

If the command executes successfully and you specified DETAIL as the *list_options* parameter value, the following parameter is returned:

plu_alias
Partner LU alias.

fqplu_name
A 17-byte fully qualified network name of the partner LU.

description
A text string describing the partner LU, as specified in the definition of the partner LU.

plu_un_name
Uninterpreted name of the partner LU (the name of the LU as defined to the remote SSCP).

max_mc_ll_send_size
The maximum size of logical records that can be sent and received by mapped conversation services at the partner LU. This value is a number in the range 1–32,767, or 0 (zero) to specify no limit (in which case the maximum is 32,767 bytes).

conv_security_ver
Specifies whether the partner LU is authorized to validate user IDs on behalf of local LUs (whether the partner LU can set the already verified indicator in an Attach request). Possible values are:

YES The partner LU can validate user IDs.

NO The partner LU cannot validate user IDs.

parallel_sess_supp
Specifies whether the partner LU supports parallel sessions. Possible values are:

YES The partner LU supports parallel sessions.

NO The partner LU does not support parallel sessions.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_PLU_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *plu_alias* or *fqplu_name* parameter value was not valid.

State Check
No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions
Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

query_port

The **query_port** command returns information about the definition of a port. If the port is active, this command also returns information about its status. This command can be used to obtain summary or detailed information about a specific port or about multiple ports, depending on the options used.

Supplied Parameters

Parameter name	Type	Length	Default
[query_port]			
num_entries	decimal		1
list_options	constant		SUMMARY + LIST_INCLUSIVE
port_name	character	8	(null string)
dlc_name	character	8	(null string)

Supplied parameters are:

num_entries
Maximum number of ports for which data should be returned. You can specify 1 to return data for a specific port, a number greater than 1 to return data for multiple ports, or 0 (zero) to return data for all ports.

list_options
The level of information required for each entry and the position in the list of ports from which Communications Server for Linux begins to return data.

Specify the level of information required with one of the following values:

SUMMARY
Summary information only

DETAIL Detailed information

Use a + character to combine this value with one of the following values:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the *port_name* parameter

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *port_name* parameter

port_name
Name of the port for which information is required, or the name to be used as an index into the list of ports. This parameter is ignored if *list_options* is set to FIRST_IN_LIST.

dlc_name
DLC name filter. To return information only on ports associated with a specific DLC, specify the DLC name. This name is an 8-byte character string. To return information about all ports without filtering on the DLC name, do not specify this parameter.

Returned Parameters: Summary Information

Parameter name	Type	Length
port_name	character	8
description	character	31
port_state	character	8
dlc_name	character	8

If the command executes successfully and you specified SUMMARY as the *list_options* parameter value, Communications Server for Linux returns the following parameters:

port_name
Name of the port.

description
A text string describing the port, as specified in the definition of the port.

port_state
Specifies the current state of the port. Possible values are:

ACTIVE The port is active.

NOT_ACTIVE
The port is not active.

PENDING_ACTIVE
The **start_port** command is in progress.

PENDING_INACTIVE
The **stop_port** command is in progress.

dlc_name
Name of the DLC associated with this port.

Returned Parameters: Detailed Information

The following information is returned for all DLC types when you specify DETAIL for the *list_options* parameter.

Parameter name	Type	Length
port_name	character	8
port_state	constant	
dlc_type	constant	
port_sim_rim	constant	
def_ls_good_xids	decimal	
def_ls_bad_xids	decimal	
dyn_ls_good_xids	decimal	
dyn_ls_bad_xids	decimal	
num_implicit_links	decimal	

```
neg_ls_supp                   constant
abm_ls_supp                   constant
start_time                    decimal
description                   character     31
dlc_name                      character     8
port_type                     constant
port_number                   decimal
max_rcv_btu_size              decimal
tot_link_act_lim              decimal
inb_link_act_lim              decimal
out_link_act_lim              decimal
ls_role                       constant
implicit_dspu_services        constant
implicit_dspu_template        character     8
implicit_ls_limit             decimal
implicit_link_lvl_error       constant
implicit_uplink_to_en         constant
act_xid_exchange_limit        decimal
nonact_xid_exchange_limit     decimal
ls_xmit_rcv_cap               constant
max_ifrm_rcvd                 decimal
target_pacing_count           decimal
max_send_btu_size             decimal
implicit_cp_cp_sess_support   constant
implicit_limited_resource     constant
implicit_hpr_support          constant
implicit_link_lvl_error       constant
implicit_deact_timer          decimal
effect_cap                    decimal
connect_cost                  decimal
byte_cost                     decimal
security                      constant
prop_delay                    constant
user_def_parm_1               decimal
user_def_parm_2               decimal
user_def_parm_3               decimal
initially_active              constant
```

For SDLC, the following parameters are included. For more information about these parameters, see “define_sdlc_port” on page 178.

```
address                       hex number
```

For QLLC, the following parameters are included. For more information about these parameters, see “define_qllc_port” on page 150.

```
address                       character    14
cud_mode                      constant
cud_match                     character    128
add_mode                      constant
add_len                       decimal
```

For Token Ring or Ethernet, the following parameters are included. For more information about these parameters, see “define_tr_port, define_ethernet_port” on page 226.

```
mac_address                   hex array
lsap_address                  hex number
window_inc_threshold          decimal
xid_timer                     decimal
xid_timer_retry               decimal
test_timeout                  decimal
test_timer_retry              decimal
ack_timeout                   decimal
p_bit_timeout                 decimal
t2_timeout                    decimal
```

```
rej_timeout                     decimal
busy_state_timeout              decimal
idle_timeout                    decimal
max_retry                       decimal
```

For MPC, available with Communications Server for Linux on System z only, no additional parameters are included.

For Enterprise Extender (HPR/IP), the following parameters are included. The parameters *lsap— determined_ip_address* are described below; for more information about the remaining parameters, see "define_ip_port" on page 79.

```
lsap                            hex number
version                         decimal
determined_ip_address           character
ip_ack_timeout                  decimal
ip_max_retry                    decimal
liveness_timeout                decimal
short_hold_mode                 constant
local_ip_interface              character  45
```

If the command executes successfully and you specified `DETAIL` as the *list_options* parameter value, Communications Server for Linux returns the following parameters:

port_name
Name of the port.

port_state
Specifies the current state of the port. Possible values are:

ACTIVE The port is active.

NOT_ACTIVE
The port is not active.

PENDING_ACTIVE
The **start_port** command is in progress.

PENDING_INACTIVE
The **stop_port** command is in progress.

dlc_type
DLC type for the port. Possible values are:

SDLC Synchronous data link control

QLLC Qualified logical link control

TR Token Ring

ETHERNET
Ethernet

MPC Multipath Channel (MPC), Communications Server for Linux on System z only

HPRIP Enterprise Extender (HPR/IP)

port_sim_rim
Specifies whether set initialization mode (SIM) and receive initialization mode (RIM) are supported. Possible values are:

YES SIM and RIM are supported.

NO SIM and RIM are not supported.

def_ls_good_xids
Total number of successful XID exchanges that have occurred on all link stations defined on this port since the last time this port was started.

def_ls_bad_xids
Total number of unsuccessful XID exchanges that have occurred on all link stations defined on this port since the last time this port was started.

dyn_ls_good_xids
Total number of successful XID exchanges that have occurred on all dynamic link stations on this port since the last time this port was started.

dyn_ls_bad_xids
Total number of unsuccessful XID exchanges that have occurred on all dynamic link stations on this port since the last time this port was started.

num_implicit_links
Total number of implicit links currently active on this port. This includes dynamic links and implicit links created following use of Discovery. The number of such links allowed on this port is limited by the *implicit_ls_limit* parameter.

neg_ls_supp
Support for negotiable link stations. Possible values are:

YES Link stations can be negotiated.

NO Link stations cannot be negotiated.

abm_ls_supp
Support for ABM link stations. Possible values are:

YES ABM link stations are supported.

NO ABM link stations are not supported.

UNKNOWN
Support for ABM link stations cannot be determined because the DLC associated with this port has not yet been started.

start_time
The elapsed time, in hundredths of a second, between the time the node was started and the last time this port was started. If this port has not yet been started, this parameter is set to zero.

description
A text string describing the port, as specified in the definition of the port.

dlc_name
Name of the DLC associated with this port.

lsap Enterprise Extender (HPR/IP): Link Service Access Point address of the port.

version Enterprise Extender (HPR/IP): IP version for which this IP address is defined. Possible values are:

IP_VERSION_4
IPv4 dotted-decimal IP address (such as `193.1.11.100`).

IP_VERSION_6
IPv6 colon-hexadecimal address (such as `2001:0db8:0000:0000:0000:0000:1428:57ab` or `2001:db8::1428:57ab`).

determined_ip_address
Enterprise Extender (HPR/IP): IP address of the local link station. This is an IPv4 dotted-decimal address (such as `193.1.11.100`) or an IPv6 colon-hexadecimal address (such as `2001:0db8:0000:0000:0000:0000:1428:57ab` or `2001:db8::1428:57ab`), as indicated by the *version* parameter above. If the port is inactive, the address appears as all zeros.

For more information about the remaining parameters, see the **define_*_port** command for the appropriate port type.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
`PARAMETER_CHECK`

secondary_rc
Possible values are:

`INVALID_PORT_NAME`
The *list_options* parameter was set to `LIST_INCLUSIVE` to list all entries starting from the supplied name, but the value specified in the *port_name* parameter was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

query_pu

The **query_pu** command returns information about local PUs and the links associated with them. This command can be used to obtain information about a specific PU or about multiple PUs, depending on the options used.

Supplied Parameters

Parameter name	Type	Length	Default
[query_pu]			
num_entries	decimal		1
list_options	constant		LIST_INCLUSIVE
pu_name	character	8	(null string)
host_attachment	constant		NONE

Supplied parameters are:

num_entries
Maximum number of PUs for which data should be returned. You can

specify 1 to return data for a specific PU, a number greater than 1 to return data for multiple PUs, or `0` (zero) to return data for all PUs.

list_options
The position in the list of PUs from which Communications Server for Linux begins to return data.

Specify one of the following values:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the *pu_name* parameter

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *pu_name* parameter

pu_name
Name of the PU for which information is required, or the name to be used as an index into the list of PUs. This value is ignored if *list_options* is set to `FIRST_IN_LIST`. This name is a type-A character string of 1–8 characters.

host_attachment
Specifies whether to filter the returned information by whether the PUs are attached to the host directly or using DLUR. Possible values are:

DIRECT_ATTACHED
Return information only about PUs directly attached to the host system.

DLUR_ATTACHED
Return information only about PUs supported by DLUR.

NONE Return information about all PUs regardless of host attachment.

Returned Parameters

Parameter name	Type	Length
pu_name	character	8
description	character	31
ls_name	character	8
pu_sscp_sess_active	constant	
host_attachment	constant	
max_send_btu_size	decimal	
max_rcv_btu_size	decimal	
send_fmd_data_frames	decimal	
rcv_fmd_data_frames	decimal	
send_data_frames	decimal	
send_data_bytes	decimal	
rcv_data_frames	decimal	
rcv_data_bytes	decimal	
sidh	hex number	
sidl	hex number	
odai	constant	
sscp_id	hex	6
conventional_lu_compression	constant	
dddlu_supported	constant	
tcpcv_supported	constant	
dddlu_offline_supported	constant	

If the command executes successfully, the following parameters are returned:

pu_name
PU Name.

description
A text string describing the PU, as specified in the definition of the LS or of the internal PU.

ls_name
Name of the link station associated with this PU.

pu_sscp_sess_active
Specifies whether the PU-SSCP session is active. Possible values are:

YES The session is active.

NO The session is inactive.

host_attachment
Local PU host attachment type.

Possible values are:

DIRECT_ATTACHED
PU is directly attached to the host system.

DLUR_ATTACHED
PU is supported by DLUR.

max_send_btu_size
Maximum BTU size that can be sent. This value includes the length of the TH and RH (total 9 bytes), as well as the RU.

max_rcv_btu_size
Maximum BTU size that can be received. This value includes the length of the TH and RH (total 9 bytes), as well as the RU.

send_fmd_data_frames
Number of normal flow FMD data frames sent.

rcv_fmd_data_frames
Number of normal flow FMD data frames received.

send_data_frames
Number of normal flow data frames sent.

send_data_bytes
Number of normal flow data bytes sent.

rcv_data_frames
Number of normal flow data frames received.

rcv_data_bytes
Number of normal flow data bytes received.

The following three parameters identify the Local Form Session Identifier (LFSID) for a session on the specified LS. The LFSID consists of the following parameters:

sidh Session ID high byte

sidl Session ID low byte

odai Origin Destination Assignor Indicator. Possible values are:

YES The BIND sender is the node containing the secondary link station.

NO The BIND sender is the node containing the primary link station.

sscp_id For dependent LU sessions, this parameter is the SSCP ID received in the ACTPU from the host for the PU to which the local LU is mapped. For

independent LU sessions, this parameter is set to 0 (zero). This value is an array of six bytes displayed as hexadecimal values.

conventional_lu_compression
Specifies whether data compression is requested for LU 0–3 sessions using this PU. Possible values are:

YES Data compression should be used for LU 0–3 sessions using this PU if the host requests it.

NO Data compression should not be used for LU 0–3 sessions using this PU.

dddlu_supported
Specifies whether the host system supports DDDLU (Dynamic Definition of Dependent LUs). Possible values are:

YES The host supports DDDLU.

NO The host does not support DDDLU.

tcpcv_supported
Specifies whether the host system supports receiving the TCP/IP Information Control Vector (0x64). Communications Server for Linux can use this vector to send TCP/IP addressing information for TN3270 or LUA clients to the host. Possible values are:

YES The host supports TCP CVs.

NO The host does not support TCP CVs.

dddlu_offline_supported
Specifies whether the local PU supports sending NMVT (power off) messages to the host. If the host system supports DDDLU (Dynamic Definition of Dependent LUs), Communications Server for Linux sends NMVT (power off) to the host when it has finished using a dynamically defined LU. This allows the host to save resources by removing the definition when it is no longer required.

Possible values are:

YES The local PU sends NMVT (power off) messages to the host.

NO The local PU does not send NMVT (power off) messages to the host.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_PU_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all

entries starting from the supplied name, but the value specified in the *pu_name* parameter was not valid.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc

`STATE_CHECK`

secondary_rc

`INVALID_PU_TYPE`
The PU specified by the *pu_name* parameter is a remote PU and not a local PU.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists additional combinations of primary and secondary return codes that are common to all commands.

query_rapi_clients

The **query_rapi_clients** command returns information about Remote API Clients (on AIX, Linux, or Windows)for which a particular server on the Communications Server for Linux LAN is currently acting as the master.

This command must be issued to a server. It does not matter whether the node is started on that server.

Note: If a client is connected to the server through a Web server, and the client software is stopped, there may be a delay of a minute or two before the Web server ends the connection to the Communications Server for Linux master server. This means that a **query_rapi_clients** command may still include the client for a short time after it has stopped using the server.

Supplied Parameters

Parameter name	Type	Length	Default
[query_rapi_clients]			
num_entries	decimal		1
list_options	constant		LIST_INCLUSIVE
sys_name	character	128	(null string)

Supplied parameters are:

num_entries
Maximum number of clients for which data should be returned. You can specify `1` to return data for a specific client, a number greater than `1` to return data for multiple clients, or `0` (zero) to return data for all clients.

list_options
The position in the list of clients from which Communications Server for Linux begins to return data. The list is ordered by client name. Possible values are:

`FIRST_IN_LIST`
Start at the first entry in the list of clients

`LIST_INCLUSIVE`
Start at the entry specified by the *sys_name* parameter

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *sys_name* parameter

sys_name
Fully-qualified system name of the client to be used as an index into the list (such as `newbox.this.co.uk`). This parameter is ignored if *list_options* is set to `FIRST_IN_LIST`.

Returned Parameters

If the command executes successfully, Communications Server for Linux returns the following parameter:

Parameter name	Type	Length
max_clients	decimal	

For each client, Communications Server for Linux returns the following parameters:

Parameter name	Type	Length
sys_name	character	128
origin_ip_addr	character	39
adj_ip_addr	character	39
port_number	decimal	

If the command executes successfully, Communications Server for Linux returns the following parameter:

max_clients
The maximum number of clients using the server as their master server at any time since the Communications Server for Linux software was started.

For each client, Communications Server for Linux returns the following parameters:

sys_name
The fully-qualified system name of the client (such as `newbox.this.co.uk`).

origin_ip_addr
The IP address of the client. This is one of the following:

- IPv4 address, specified as a dotted-decimal address (such as `193.1.11.100`).
- IPv6 address, specified as a colon-hexadecimal address (such as `2001:0db8:0000:0000:0000:0000:1428:57ab` or `2001:db8::1428:57ab`).

adj_ip_addr
The IP address through which the client attaches to Communications Server for Linux. This may not be the same as *rapi_client_origin_ip_addr* if one of the following is true.

- The client connects through a Web server.
- The client connects through a TCP/IP proxy or NAT router, such as the Linux iptables tool.
- The client has multiple IP addresses.

The IP address is one of the following:

- IPv4 address, specified as a dotted-decimal address (such as `193.1.11.100`).
- IPv6 address, specified as a colon-hexadecimal address (such as `2001:0db8:0000:0000:0000:0000:1428:57ab` or `2001:db8::1428:57ab`).

port_number
The IP port number through which the client attaches to Communications Server for Linux.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_NODE_NAME
The *list_options* parameter was set to LIST_INCLUSIVE or LIST_FROM_NEXT to list all entries starting from the supplied node name, but the *sys_name* parameter was not specified or was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

query_rcf_access

The **query_rcf_access** command returns information about the permitted access to the Communications Server for Linux Remote Command Facility (RCF): the user ID used to run UNIX Command Facility (UCF) commands, and the restrictions under which administration commands can be issued using the Service Point Command Facility (SPCF). This information was previously defined using **define_rcf_access**. For more information about SPCF and UCF, refer to the *Communications Server for Linux Administration Guide*.

Because RCF access parameters are defined as domain resources, this command is not associated with a particular node.

Supplied Parameters

```
[query_rcf_access]
```

No parameters are supplied for this command.

Returned Parameters

Parameter name	Type	Length
ucf_username	character	31
spcf_permissions	constant	

If the command executes successfully, Communications Server for Linux returns the following parameters:

ucf_username
Specifies the Linux user name of the UCF user. All UCF commands are run using the user ID for this user, with the default shell and access permissions defined for this user.

If this parameter is not set, UCF access is denied.

spcf_permissions
Specifies the types of Communications Server for Linux administration commands that can be accessed using SPCF. To prevent access to SPCF, set this parameter to `NONE`. To allow access to SPCF, set this parameter to one or more of the following values (combined using a + character):

ALLOW_QUERY_LOCAL
The **query**_* commands are allowed.

ALLOW_DEFINE_LOCAL
The **define**_*, **set**_*, **delete**_*, **add**_*, **remove**_*, and **init_node** commands are allowed.

ALLOW_ACTION_LOCAL
The **start**_*, **stop**_*, **activate**_*, **deactivate**_*, **aping**, **initialize_session_limit**, **change_session_limit**, and **reset_session_limit** commands are allowed.

ALLOW_QUERY_REMOTE
The **query**_* commands are allowed to provide access to a remote Communications Server for Linux node.

ALLOW_DEFINE_REMOTE
The **define**_*, **set**_*, **delete**_*, **add**_*, **remove**_*, and **init_node** commands are allowed to provide access to a remote Communications Server for Linux node.

ALLOW_ACTION_REMOTE
The **start**_*, **stop**_*, **activate**_*, **deactivate**_*, **aping**, **initialize_session_limit**, **change_session_limit**, and **reset_session_limit** commands are allowed to provide access to a remote Communications Server for Linux node.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_rtp_connection

The **query_rtp_connection** command returns a list of information about Rapid Transport Protocol (RTP) connections for which the node is an endpoint. RTP is a High Performance Routing (HPR) protocol, supported by network nodes only, that enables a network node endpoint to set up an APPN HPR connection that offers better data routing performance and session reliability than is available on APPN intermediate session routing (ISR) connections.

This command can be used to obtain summary or detailed information about a specific RTP connection or about multiple RTP connections, depending on the options used.

This command must be issued to a running node.

Supplied Parameters

Parameter name	Type	Length	Default
[query_rtp_connection]			
num_entries	decimal		1
list_options	constant		SUMMARY+LIST_INCLUSIVE
rtp_name	character	8	(null string)

Supplied parameters are:

num_entries
: Maximum number of RTP connections for which data should be returned. You can specify 1 to return data for a specific RTP connection, a number greater than 1 to return data for multiple RTP connections, or 0 to return data for all RTP connections.

list_options
: The level of information required for each entry and the position in the list from which Communications Server for Linux begins to return data.

  Specify the level of information required with one of the following values:

  SUMMARY
  : Summary information only.

  DETAIL Detailed information.

  Use a + character to combine this value with one of the following values:

  FIRST_IN_LIST
  : Start at the first entry in the list.

  LIST_INCLUSIVE
  : Start at the entry specified by the *rtp_name* parameter.

  LIST_FROM_NEXT
  : Start at the entry immediately following the entry specified by the *rtp_name* parameter.

rtp_name
: Name of the RTP connection. This value is ignored if the *list_options* parameter is set to FIRST_IN_LIST.

Returned Parameters: Summary Information

Parameter	Type	Length
rtp_name	character	8
first_hop_ls_name	character	8

dest_node_name	character	17
cos_name	character	8
num_sess_active	decimal	
connection_type	constant	

If the command executes successfully and you specified SUMMARY as the *list_options* parameter value, Communications Server for Linux returns the following parameters:

rtp_name
Name of the RTP connection.

first_hop_ls_name
Name of the link station of the first hop of the RTP connection.

dest_node_name
Fully qualified name of the destination control point for the RTP portion of the session.

cos_name
Name of the class of service used by the RTP connection.

num_sess_active
Number of sessions active on this RTP connection.

connection_type
Specifies the type of sessions on the RTP connection. Possible values are:

CP_CP_SESSION
The RTP connection carries CP-CP sessions.

LU_LU_SESSION
The RTP connection carries LU-LU sessions.

ROUTE_SETUP
The RTP connection is used for route setup.

Returned Parameters: Detailed Information

Parameter	Type	Length
rtp_name	character	8
first_hop_ls_name	character	8
dest_node_name	character	17
cos_name	character	8
num_sess_active	decimal	
arb_mode	constant	
connection_type	constant	
max_btu_size	decimal	
liveness_timer	decimal	
local_tcid	hex array	8
remote_tcid	hex array	8
bytes_sent	decimal	
bytes_received	decimal	
bytes_resent	decimal	
bytes_discarded	decimal	
packets_sent	decimal	
packets_received	decimal	
packets_resent	decimal	
packets_discarded	decimal	
gaps_detected	decimal	
send_rate	decimal	
max_send_rate	decimal	
min_send_rate	decimal	
receive_rate	decimal	
max_receive_rate	decimal	
min_receive_rate	decimal	

```
burst_size                  decimal
up_time                     decimal
smooth_rtt                  decimal
last_rtt                    decimal
short_req_timer             decimal
short_req_timeouts          decimal
liveness_timeouts           decimal
in_invalid_sna_frames       decimal
in_sc_frames                decimal
out_sc_frames               decimal
delay_change_sum            decimal
current_receiver_threshold  decimal
minimum_receiver_threshold  decimal
maximum_receiver_threshold  decimal
sent_normals_count          decimal
sent_slowdowns_count        decimal
rcvd_normals_count          decimal
rcvd_slowdowns_count        decimal
dcs_reset_count_non_heal    decimal
dcs_reset_count_healing     decimal
arb_mode_color              decimal
route                       character
```

If the command executes successfully and you specified DETAIL as the *list_options* parameter value, Communications Server for Linux returns the following parameters:

rtp_name
Name of the RTP connection.

first_hop_ls_name
Name of the link station of the first hop of the RTP connection.

dest_node_name
Fully qualified name of the destination control point for the RTP portion of the session.

cos_name
Name of the class of service used by the RTP connection.

num_sess_active
Number of sessions active on this RTP connection.

arb_mode
Specifies the ARB mode in use on this RTP Connection. Possible values are:

ARB_S Standard mode ARB.

ARB_R Responsive mode ARB.

UNKNOWN
The ARB mode has not yet been determined because the RTP connection is not yet established.

connection_type
Specifies the type of sessions on the RTP connection. Possible values are:

RTP_CP_CP_SESSION
The RTP connection carries CP-CP sessions.

RTP_LU_LU_SESSION
The RTP connection carries LU-LU sessions.

RTP_ROUTE_SETUP
The RTP connection is used for route setup.

max_btu_size
Maximum size, in bytes, of the basic transmission unit (BTU) used on the RTP connection.

liveness_timer
Liveness timer, measured in seconds, for the RTP connection. If no traffic flows on a connection during a liveness timer interval, RTP starts a status exchange to check if its partner is still there. A short liveness timer interval provides quick detection of line failures and rapid path switching when a line fails. However, if the interval is too short, performance is slightly degraded by the frequent checks on the status of the line.

local_tcid
Local TCID (transport control identifier) for the RTP connection.

remote_tcid
Remote TCID for the RTP connection.

bytes_sent
Total number of bytes that the local node has sent on this RTP connection.

bytes_received
Total number of bytes that the local node has received on this RTP connection.

bytes_resent
Total number of bytes that the local node has resent on this RTP connection because bytes were lost in transit.

bytes_discarded
Total number of bytes sent by the other end of the RTP connection that were discarded as duplicates of data already received.

packets_sent
Total number of packets that the local node has sent on this RTP connection.

packets_received
Total number of packets that the local node has received on this RTP connection.

packets_resent
Total number of packets that the local node has resent on this RTP connection because packets were lost in transit.

packets_discarded
Total number of packets sent by the other end of the RTP connection that were discarded as duplicates of data already received.

gaps_detected
Total number of gaps detected by the local node. Each gap corresponds to one or more lost frames.

send_rate
Current send rate on this RTP connection, measured in Kbits/second. This rate is the maximum allowed send rate as calculated by the ARB (adaptive rate-based) algorithm. RTP uses the ARB algorithm to calculate how fast it can send data based on an analysis of the amount of time it takes for the partner to respond.

max_send_rate
Maximum send rate on this RTP connection, measured in Kbits/second.

min_send_rate
Minimum send rate on this RTP connection, measured in Kbits/second.

receive_rate
Current receive rate on this RTP connection, measured in Kbits/second. This rate is the actual rate calculated over the last measurement interval.

max_receive_rate
Maximum receive rate on this RTP connection, measured in Kbits/second.

min_receive_rate
Minimum receive rate on this RTP connection, measured in Kbits/second.

burst_size
Current burst size on this RTP connection, measured in bytes.

up_time
Total number of seconds this RTP connection has been active.

smooth_rtt
Smoothed measure of round-trip time between the local node and the partner RTP node, measured in milliseconds.

last_rtt The last measured round-trip time between the local node and the partner RTP node, measured in milliseconds.

short_req_timer
The amount of time to wait for a response to a request for a status exchange, measured in milliseconds. A short timer interval provides quick detection of failures but lowers performance.

short_req_timeouts
Total number of times the *short_req_ timer* has expired for this RTP connection.

liveness_timeouts
Total number of times the liveness timer has expired for this RTP connection. The liveness timer expires when the connection has been idle for the period specified in the *liveness_timer* parameter.

in_invalid_sna_frames
Total number of SNA frames received and discarded on this RTP connection because they were not valid.

in_sc_frames
Total number of session control frames received on this RTP connection.

out_sc_frames
Total number of session control frames sent on this RTP connection.

delay_change_sum
Value of the delay change sum currently held by the ARB-R algorithm on this RTP connection.

current_receiver_threshold
Value of the receiver threshold currently held by the ARB-R algorithm on this RTP connection.

minimum_receiver_threshold
Value of the minimum receiver threshold currently held by the ARB-R algorithm on this RTP connection.

maximum_receiver_threshold
Value of the maximum receiver threshold currently held by the ARB-R algorithm on this RTP connection.

sent_normals_count
Number of NORMAL feedback ARB-R segments sent by the ARB-R algorithm on this RTP connection.

sent_slowdowns_count
Number of SLOWDOWN1 and SLOWDOWN2 feedback ARB-R segments sent by the ARB-R algorithm on this RTP connection.

rcvd_normals_count
Number of NORMAL feedback ARB-R segments received by the ARB-R algorithm on this RTP connection.

rcvd_slowdowns_count
Number of SLOWDOWN1 and SLOWDOWN2 feedback ARB-R segments received by the ARB-R algorithm on this RTP connection.

dcs_reset_count_non_heal
Number of delay change sum resets made as a part of normal ARB-R processing on this RTP connection.

dcs_reset_count_healing
Number of delay change sum resets made to self-heal the ARB-R algorithm on this RTP connection.

arb_mode_color
The current ARB-R status mode on this RTP connection. Possible values are:

0 GREEN

1 YELLOW

2 RED

route Route Selection Control Vector (RSCV) as defined by SNA Formats. This control vector defines the session route through the network and is carried on the BIND. This RSCV is included only if the node's configuration (specified using **define_node**) indicates that endpoint RSCVs should be stored.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_RTP_CONNECTION
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *rtp_name* parameter was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

query_rtp_tuning

The **query_rtp_tuning** command returns information about the parameters that will be used for future RTP connections. This information was previously set up using **define_rtp_tuning**.

Supplied Parameters

```
[query_rtp_tuning]
```

No parameters are supplied for this command.

Returned Parameters

```
Parameter name            Type           Length
path_switch_attempts      decimal
short_req_retry_limit     decimal
path_switch_time_low      decimal
path_switch_time_medium   decimal
path_switch_time_high     decimal
path_switch_time_network  decimal
refifo_cap                decimal
srt_cap                   decimal
```

If the command executes successfully, Communications Server for Linux returns the following parameters:

path_switch_attempts
: Number of path switch attempts to set on new RTP connections.

short_req_retry_limit
: Number of times a Status Request is sent before Communications Server for Linux determines that an RTP connection is disconnected and starts Path Switch processing.

path_switch_time_low
: Length of time in seconds for which Communications Server for Linux attempts to path switch a disconnected RTP connection with transmission priority `AP_LOW`.

path_switch_time_medium
: Length of time in seconds for which Communications Server for Linux attempts to path switch a disconnected RTP connection with transmission priority `AP_MEDIUM`.

path_switch_time_high
: Length of time in seconds for which Communications Server for Linux attempts to path switch a disconnected RTP connection with transmission priority `AP_HIGH`.

path_switch_time_network
Length of time in seconds for which Communications Server for Linux attempts to path switch a disconnected RTP connection with transmission priority AP_NETWORK.

refifo_cap
The RTP protocol uses a timer called the Re-FIFO Timer. The value of this timer is calculated as part of the protocol, but this parameter specifies a maximum value in milliseconds beyond which the timer cannot increase. In some situations, setting this maximum value can improve performance. A value of 0 (zero) means that the timer is not limited and can take any value calculated by the protocol.

srt_cap The RTP protocol uses a timer called the Short Request Timer. The value of this timer is calculated as part of the protocol, but this parameter specifies a maximum value in milliseconds beyond which the timer cannot increase. In some situations, setting this maximum value can improve performance. A value of 0 (zero) means that the timer is not limited and can take any value calculated by the protocol.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_security_access_list

The **query_security_access_list** command returns information about security access lists defined in a Communications Server for Linux configuration file. It can return information about a single list or multiple lists, depending on the options used.

Supplied Parameters

Parameter name	Type	Length	Default
[query_security_access_list]			
num_entries	decimal		1
list_options	constant		LIST_INCLUSIVE
list_name	character	14	(null string)
user_name	character	10	(null string)

Supplied parameters are:

num_entries
Maximum number of security access lists for which data should be

returned. You can specify 1 to return data for a specific list, a number greater than 1 to return data for multiple lists, or 0 (zero) to return data for all lists.

This number includes partial security access list entries (for which a user name is specified, so that the returned data does not include the first user name in the list).

list_options
The position in the list from which Communications Server for Linux begins to return data. Specify one of the following values:

FIRST_IN_LIST
Start at the first user name for the first security access list

LIST_INCLUSIVE
Start at the entry specified by the supplied security access list name and user name, or start at the first user name for the specified security access list if no user name is specified

LIST_FROM_NEXT
If a user name is specified, start at the user immediately following the specified user. If no user name is specified, start at the first user for the specified security access list.

list_name
The name of the security access list for which information is required, or the name to be used as an index into the list of security access lists. This parameter is ignored if *list_options* is set to FIRST_IN_LIST. The name is a character string of 1–14 locally displayable characters.

user_name
To return information starting with a specific user name for the specified security access list, set this parameter to the user name. To return information starting at the first user name for the specified security access list, do not specify this parameter.

Returned Parameters

Parameter name	Type	Length
list_name	character	14
description	character	31
num_users	decimal	
{security_user_data}		
user_name	character	10

If the command executes successfully, the following parameters are returned:

list_name
The name of the security access list.

description
An optional string of 0–31 characters.

num_users
Number of user names in the list.

For each user name in the list, a security_user_data subrecord is returned with the following information:

user_name
Name of the user.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_LIST_NAME
The *list_options* parameter was set to LIST_INCLUSIVE, but the *list_name* parameter did not match the name of any defined security access list.

INVALID_USER_NAME
The *list_options* parameter was set to LIST_INCLUSIVE, but the *user_name* parameter did not match a user name defined for the specified security access list.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

query_session

The **query_session** command returns information about sessions for a particular local LU. This command can be used to obtain summary or detailed information about a specific session or a range of sessions, depending on the options used.

This command must be issued to a running node.

Supplied Parameters

Parameter name	Type	Length	Default
[query_session]			
num_entries	decimal		1
list_options	constant		SUMMARY + LIST_INCLUSIVE
lu_name	character	8	(null string)
lu_alias	character	8	(null string)
plu_alias	character	8	(null string)
fqplu_name	character	17	(null string)
mode_name	character	8	(null string)
session_id	hex array	8	

Supplied parameters are:

num_entries
Maximum number of sessions for which data should be returned. You can

specify 1 to return data for a specific session, a number greater than 1 to return data for multiple sessions, or 0 to return data for all sessions.

list_options
The level of information required for each entry and the position in the list of sessions from which Communications Server for Linux begins to return data.

Specify the level of information required with one of the following values:

SUMMARY
Summary information only

DETAIL Detailed information

Use a + character to combine this value with one of the following values:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the combination of local LU, partner LU, mode name, and session ID

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the combination of local LU, partner LU, mode name, and session ID

lu_name
LU name of the local LU. This name is a type-A character string. To indicate that the LU is identified by its LU alias instead of its LU name, do not specify this parameter. To specify the LU associated with the local CP (the default LU), do not specify either *lu_name* or *lu_alias*.

lu_alias
Locally defined LU alias. This parameter is used only if *lu_name* is not specified. To specify the LU associated with the local CP (the default LU), do not specify either *lu_name* or *lu_alias*.

plu_alias
Partner LU alias. To return information only about sessions associated with a specific partner LU, specify the partner LU alias or the partner LU fully qualified network name (*fqplu_name*). To return information about all sessions without filtering on the partner LU, do not specify either of these parameters.

To specify that the LU is identified by its LU name rather than its alias, specify *fqplu_name* and not *plu_alias*.

fqplu_name
Fully qualified network name of the partner LU. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character partner LU name. For the network name and partner LU name, use only letters, digits 0–9, and special characters $, #, and @.

This parameter is used only if the *plu_alias* parameter is not specified; it is ignored if *plu_alias* is specified.

mode_name
Mode name. This name is a type-A character string. To return information only about sessions associated with a specific mode, specify the mode

name; the partner LU must also be specified (using *plu_alias* or *fqplu_name*). To return information about all sessions without filtering on mode name, do not specify this parameter.

session_id
The 8-byte identifier of the session for which information is required, or the session ID to be used as an index into the list of sessions. This parameter is ignored if *list_options* is set to FIRST_IN_LIST.

Returned Parameters: Summary Information

Parameter name	Type	Length
plu_alias	character	8
fqplu_name	character	17
mode_name	character	8
session_id	hex array	8
pcid	hex array	8
fqcp_name	character	17

If the command executes successfully and you specified SUMMARY as the *list_options* parameter value, Communications Server for Linux returns the following parameters:

plu_alias
Partner LU alias.

fqplu_name
A 17-byte fully qualified network name of the partner LU.

mode_name
Mode name.

session_id
An 8-byte identifier of the session.

pcid Procedure Correlator ID.

fqcp_name
Fully qualified CP name of the node.

Returned Parameters: Detailed Information

Parameter name	Type	Length
plu_alias	character	8
fqplu_name	character	17
mode_name	character	8
session_id	hex array	8
pcid	hex array	8
fqcp_name	character	17
cos_name	character	8
trans_pri	constant	
ltd_res	constant	
polarity	constant	
contention	constant	
rcv_ru_size	decimal	
send_ru_size	decimal	
max_send_btu_size	decimal	
max_rcv_btu_size	decimal	
max_send_pac_win	decimal	
cur_send_pac_win	decimal	
max_rcv_pac_win	decimal	
cur_rcv_pac_win	decimal	
send_data_frames	decimal	
send_fmd_data_frames	decimal	
send_data_bytes	decimal	

```
rcv_data_frames         decimal
rcv_fmd_data_frames     decimal
rcv_data_bytes          decimal
sidh                    hex number
sidl                    hex number
odai                    constant
ls_name (or rtp_name)   character      8
pacing_type             constant
duplex_support          constant
sscp_id                 decimal
session_start_time      decimal
session_timeout         decimal
plu_slu_comp_level      constant
slu_plu_comp_level      constant
```

If the command executes successfully and you specified DETAIL as the *list_options* parameter value, Communications Server for Linux returns the following parameters:

plu_alias
Partner LU alias.

fqplu_name
A 17-byte fully qualified network name of the partner LU.

mode_name
Mode name.

session_id
An 8-byte identifier of the session.

pcid Procedure Correlator ID.

fqcp_name
Fully qualified CP name of the node.

cos_name
Class of service name.

trans_pri
Transmission priority. Possible values are:

LOW Low priority is given to the transmission.

MEDIUM Medium priority is given to the transmission.

HIGH High priority is given to the transmission.

NETWORK
Highest priority is given to the transmission.

ltd_res Specifies whether the session uses a limited resource link. Possible values are:

YES Session uses a limited resource link.

NO Session does not use a limited resource link.

polarity
Specifies the polarity of the session. Possible values are:

PRIMARY
Primary polarity

SECONDARY
Secondary polarity

contention
Specifies whether the session is a contention winner or contention loser session for the local LU. Possible values are:

CONWINNER
Contention winner session

CONLOSER
Contention loser session

rcv_ru_size
Maximum RU size that can be received.

send_ru_size
Maximum RU size that can be sent.

max_send_btu_size
Maximum BTU size that can be sent.

max_rcv_btu_size
Maximum BTU size that can be received.

max_send_pac_win
Maximum size of the send pacing window on this session.

cur_send_pac_win
Current size of the send pacing window on this session.

max_rcv_pac_win
Maximum size of the receive pacing window on this session.

cur_rcv_pac_win
Current size of the receive pacing window on this session.

send_data_frames
Number of normal flow data frames sent.

send_fmd_data_frames
Number of normal flow FMD data frames sent.

send_data_bytes
Number of normal flow data bytes sent.

rcv_data_frames
Number of normal flow data frames received.

rcv_fmd_data_frames
Number of normal flow FMD data frames received.

rcv_data_bytes
Number of normal flow data bytes received.

The following three parameters identify the Local Form Session Identifier (LSFID):

sidh Session ID high byte

sidl Session ID low byte

odai Origin Destination Assignor Indicator. Possible values are:

YES The BIND sender is the node containing the secondary link station.

NO The BIND sender is the node containing the primary link station.

ls_name
Link station name associated with statistics. This can be used to correlate the session statistics with the link over which session data flows.

This parameter is not included if the session uses a Rapid Transport Protocol (RTP) connection; the *rtp_name* parameter is used instead.

rtp_name
Name of the Rapid Transport Protocol (RTP) connection used by the session.

This parameter is not included if the session does not use an RTP connection; the *ls_name* parameter is used instead.

pacing_type
The type of receive pacing in use on this session. Possible values are:

```
NONE
FIXED
ADAPTIVE
```

duplex_support
Returns the conversation duplex support as negotiated on the BIND. Possible values are:

HALF-DUPLEX
Only half-duplex conversations are supported.

FULL_DUPLEX
Both full-duplex and half-duplex sessions are supported. Expedited data is also supported.

sscp_id For dependent LU sessions, this parameter is the SSCP ID received in the ACTPU from the host for the PU to which the local LU is mapped. For independent LU sessions, this parameter is set to `0` (zero).

session_start_time
The time between the CP starting and this session becoming active, measured in one-hundredths of a second. If the session is not fully active when the query is processed, this parameter is set to `0` (zero).

session_timeout
The timeout associated with this session. This timeout is derived from:

- The LU 6.2 timeout associated with the local LU
- The LU 6.2 timeout associated with the remote LU
- The mode timeout
- The global timeout
- The limited resource timeout (if this session is running over a limited resource link)

plu_slu_comp_lvl
Specifies the compression level for data sent from the primary LU (PLU) to the secondary LU (SLU). Possible values are:

NONE Compression is not used.

RLE Run-length encoding (RLE) compression is used.

LZ9 LZ9 compression is used.

LZ10 LZ10 compression is used.

slu_plu_comp_lvl
Specifies the compression level for data sent from the secondary LU (SLU) to the primary LU (PLU). Possible values are:

NONE Compression is not used.

RLE Run-length encoding (RLE) compression is used.

LZ9 LZ9 compression is used.

LZ10 LZ10 compression is used.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_LU_ALIAS
The *lu_alias* parameter value was not valid.

INVALID_LU_NAME
The *lu_name* parameter value was not valid.

INVALID_SESSION_ID
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *session_id* parameter value was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_sna_net

The **query_sna_net** command returns information about servers that can act as backup master servers, as defined in the **sna.net** file. This command can be used to obtain information about a specific server or about multiple servers, depending on the options used.

The order of server names in this file is significant; the first server listed in the file will always be the master if it is active, the second will be the master if the first is inactive, the third will be the master if the first and second are both inactive, and so on. Because of this ordering, the list of server names returned on **query_sna_net** is in the same order as it is in the file; the returned names are not ordered by name length and lexicographical ordering, as with other **query**_* commands.

This command must be issued without specifying a node name.

Supplied Parameters

Parameter name	Type	Length	Default
[query_sna_net]			
num_entries	decimal		1
list_options	constant		LIST_INCLUSIVE
server_name	character	128	(null string)

Supplied parameters are:

num_entries
: Maximum number of server names for which data should be returned. You can specify 1 to return data for a specific server name, a number greater than 1 to return data for multiple server names, or 0 to return data for all server names.

list_options
: The position in the list of server names from which Communications Server for Linux begins to return data. The server names are listed in the same order as in the file, not in order of name length, lexicographical order, or both, as for other **query**_* commands.

  Possible values are:

  FIRST_IN_LIST
  : Start at the first entry in the list

  LIST_INCLUSIVE
  : Start at the entry specified by the *server_name* parameter

  LIST_FROM_NEXT
  : Start at the entry immediately following the entry specified by the *server_name* parameter

server_name
: Name of the server for which information is required, or the name to be used as an index into the list of servers. The server name is ignored if *list_options* is set to FIRST_IN_LIST.

  If the computer name includes a . (period) character, Communications Server for Linux assumes that it is a fully-qualified name; otherwise it performs a DNS lookup to determine the computer name.

Returned Parameters

Parameter name	Type	Length
security	constant	
domain_name	character	64
server_name	character	128

If the command executes successfully, Communications Server for Linux returns the following parameters:

security
: This parameter is reserved.

domain_name
: The name of the TCP/IP domain containing the Communications Server for Linux domain. This name was specified during installation of the master server.

  For each server, the following parameter is included:

server_name
Name of the server listed in the file.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc

RECORD_NOT_FOUND
The *list_options* parameter was set to LIST_INCLUSIVE or LIST_FROM_NEXT to list entries starting from the supplied server name, but the *server_name* parameter did not match an entry in the file.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

query_statistics

The **query_statistics** command returns statistics on the usage of an LS or port. The MPC link type does not support link statistics; do not issue this command for an MPC LS or port. The QLLC link type does not support link statistics; do not issue this command for a QLLC LS or port.

The type of information returned depends on the DLC type:

For SDLC, the command returns either statistics (counts of events such as particular frame types sent or received) or operational information (details of parameters currently being used), for either an LS or a port.

For Token Ring or Ethernet, the command returns statistics information for either an LS or a port.

For Enterprise Extender, the verb returns statistics information for an LS.

This command must be issued to a running node.

Supplied Parameters

Parameter name	Type	Length	Default
[query_statistics]			
name	character	8	
stats_type	constant		LS

table_type	constant	STATS
reset_stats	constant	NO
dlc_type	constant	SDLC

Supplied parameters are:

name Name of the LS or port for which statistics are required.

stats_type
The type of resource for which statistics are required.

Possible values for Token Ring / Ethernet are:

LS Return LS statistics.

PORT Return port statistics.

For Enterprise Extender, this must be set to AP_LS.

table_type
The type of statistics information requested.

Possible values for SDLC are:

STATS Statistical information.

OPER Operational information.

For Token Ring / Ethernet , this parameter must be set to STATS.

For Enterprise Extender, this must be set to STATS.

reset_stats
Specifies whether to reset the statistics when this command completes. This parameter applies only if *table_type* is set to STATS; it is ignored otherwise. Possible values are:

YES Reset the statistics; a subsequent **query_statistics** command will contain only data gathered after this command was issued.

NO Do not reset the statistics; the data on this command will be included in the data returned to a subsequent **query_statistics** command.

dlc_type
Type of the DLC. Possible values are:

SDLC Synchronous data link control

TR Token Ring

ETHERNET
Ethernet

X25 X.25 packet switching

HPRIP Enterprise Extender (also known as HPR/IP)

Returned Parameters: SDLC LS Statistics

Parameter name	Type
index	decimal
address	decimal
blus_in	decimal
blus_out	decimal
octets_in	decimal
octets_out	decimal
polls_out	decimal
poll_rsps_out	decimal

```
local_busies            decimal
remote_busies           decimal
iframes_in              decimal
iframes_out             decimal
retransmits_in          decimal
retransmits_out         decimal
ioctets_in              decimal
ioctets_out             decimal
uiframes_in             decimal
uiframes_out            decimal
xids_in                 decimal
xids_out                decimal
tests_in                decimal
tests_out               decimal
rejs_in                 decimal
rejs_out                decimal
frmrs_in                decimal
frmrs_out               decimal
sims_in                 decimal
sims_out                decimal
rims_in                 decimal
rims_out                decimal
snrm_in                 decimal
snrm_out                decimal
dm_in                   decimal
dm_out                  decimal
disc_in                 decimal
disc_out                decimal
ua_in                   decimal
ua_out                  decimal
```

If the command executes successfully, the following parameters are returned:

index The index value used internally by Communications Server for Linux to identify the port that owns this LS.

address The poll address of the secondary link station.

blus_in The total number of basic link units (frames) received from the adjacent link station.

blus_out
The total number of basic link units (frames) transmitted to the adjacent link station.

octets_in
The total number of bytes, not including frame check sequences (FCSs), received from the adjacent link station.

octets_out
The total number of bytes, not including FCSs, transmitted to the adjacent link station.

polls_out
The total number of polls sent to the adjacent link station.

poll_rsps_out
The total number of polls responded to by the adjacent link station.

local_busies
The total number of times the local link station has entered the receive not ready busy state (RNR).

remote_busies
The total number of times the remote link station has entered the busy state of receive not ready (RNR).

iframes_in
The total number of I-frames received from the adjacent link station (including retries and out-of-order frames).

iframes_out
The total number of I-frames transmitted to the adjacent link station (including retries and out-of-order frames).

retransmits_in
The total number of retransmissions of I-frames from the adjacent link station.

retransmits_out
The total number of retransmissions of I-frames to the adjacent link station.

ioctets_in
The total number of bytes in I-frames received from the adjacent link station.

ioctets_out
The total number of bytes in I-frames transmitted to the adjacent link station.

uiframes_in
The total number of UI frames received from the adjacent link station.

uiframes_out
The total number of UI frames transmitted to the adjacent link station.

xids_in The total number of XID frames received from the adjacent link station.

xids_out
The total number of XID frames transmitted to the adjacent link station.

tests_in
The total number of TEST frames, commands, or responses received from the adjacent link station.

tests_out
The total number of TEST frames, commands, or responses transmitted to the adjacent link station.

rejs_in The total number of REJ frames received from the adjacent link station.

rejs_out
The total number of REJ frames transmitted to the adjacent link station.

frmrs_in
The total number of FRMR frames received from the adjacent link station.

frmrs_out
The total number of FRMR frames transmitted to the adjacent link station.

sims_in
The total number of SIM frames received from the adjacent link station.

sims_out
The total number of SIM frames transmitted to the adjacent link station.

rims_in
The total number of RIM frames received from the adjacent link station.

rims_out
The total number of RIM frames transmitted to the adjacent link station.

snrm_in
The total number of SNRM frames received from the adjacent link station.

snrm_out
The total number of SNRM frames transmitted to the adjacent link station.

dm_in The total number of DM frames received from the adjacent link station.

dm_out
The total number of DM frames transmitted to the adjacent link station.

disc_in The total number of DISC frames received from the adjacent link station.

disc_out
The total number of DISC frames transmitted to the adjacent link station.

ua_in The total number of UA frames received from the adjacent link station.

ua_out The total number of UA frames transmitted to the adjacent link station.

Returned Parameters: SDLC LS Operational Information

Parameter name	Type	Length
index	decimal	
address	decimal	
role	constant	
state	decimal	
maxdata	decimal	
replyto	decimal	
maxin	decimal	
maxout	decimal	
modulo	constant	
retries_m	decimal	
retries_t	decimal	
retries_n	decimal	
rnrlimit	decimal	
datmode	constant	
last_fail_ctrl_in	hex array	2
last_fail_ctrl_out	hex array	2
last_fail_frmr_info	hex array	5
last_fail_replyto_s	decimal	
g_poll	decimal	
sim_rim	constant	
xmit_rcv_cap	constant	

If the command executes successfully, the following parameters are returned:

index The Index value used internally by Communications Server for Linux to identify the port that owns this LS.

address The poll address of the secondary link station.

role The link role of the LS. Possible values are:

PRIMARY
This link station is defined as a primary link station.

SECONDARY
This link station is defined as a secondary link station.

NEGOTIABLE
This link station is defined as a negotiable link station.

state The internal value indicating the processing state of the LS software (for use by support personnel).

maxdata
The current maximum protocol data unit (PDU) size, including the transmission header (TH) and request header (RH), allowed for the logical link. For a switched line, this value can be negotiated during XID exchange.

replyto The current reply timeout, in hundredths of a second. This parameter applies only if the LS role is primary; its value is undefined if the LS role is secondary.

maxin The maximum number of frames that the LS can receive before it must send an acknowledgment.

maxout
The maximum number of frames that the LS can send before it must wait for an acknowledgment.

modulo The sequence number modulus for the LS. Possible values are:

EIGHT A value of 8

ONETWENTYEIGHT
A value of 128

retries_m
The maximum number of frames in a retry sequence (a sequence of frames that the LS retransmits because it has not received a positive acknowledgment for them).

retries_t
The timeout between retransmissions of a retry sequence.

retries_n
The number of times that the LS attempts to retransmit a retry sequence.

rnrlimit
The maximum length of time that the adjacent LS can remain in RNR state before the local LS considers it to be inoperative.

datmode
The communications mode used with the adjacent LS. Possible values are:

HALF Two-way alternate (half-duplex)

FULL Two-way simultaneous (full-duplex)

last_fail_ctrl_in
The control field from the last frame received before the last failure. If the LS has not LS failed, this parameter is set to zeros.

last_fail_ctrl_out
The control field from the last frame sent before the last LS failure. If the LS has not failed, this parameter is set to zeros.

last_fail_frmr_info
If the last LS failure was caused by an FRMR frame that was not valid, this parameter contains the information from the FRMR frame. If the LS has not failed or if the failure was not caused by a frame that was not valid, this parameter is set to zeros.

last_fail_replyto_s
The number of times that the reply timeout expired before the last failure. If the LS has not failed, this parameter is set to `0`.

g_poll The group poll address for the LS. If the LS is not in a group, this parameter is set to 0.

sim_rim
Specifies whether the LS supports transmission of SIM and RIM control frames. Possible values are:

YES LS supports SIM and RIM.

NO LS does not support SIM and RIM.

xmit_rcv_cap
Specifies the transmit / receive capability of the LS. Possible values are:

HALF Half-duplex

FULL Full-duplex

Returned Parameters: SDLC Port Statistics

Parameter name	Type
index	decimal
dwarf_frames	decimal
polls_out	decimal
poll_rsps_out	decimal
local_busies	decimal
remote_busies	decimal
iframes_in	decimal
iframes_out	decimal
octets_in	decimal
octets_out	decimal
protocol_errs	decimal
activity_to_s	decimal
rnrlimit_s	decimal
retries_exps	decimal
retransmits_in	decimal
retransmits_out	decimal

If the command executes successfully, the following parameters are returned:

index The index value used internally by Communications Server for Linux to identify the port.

dwarf_frames
The number of frames received by the port that were too short to be valid.

polls_out
The total number of polls sent to adjacent link stations.

poll_rsps_out
The total number of polls responded to by adjacent link stations.

local_busies
The total number of times the local link station has entered the receive not ready busy state (RNR).

remote_busies
The total number of times remote link stations have entered the receive not ready busy state (RNR).

iframes_in
The total number of I-frames received from adjacent link stations (including retries and out-of-order frames).

iframes_out
The total number of I-frames transmitted to adjacent link stations (including retries and out-of-order frames).

octets_in
The total number of bytes (not including FCSs) received from adjacent link stations.

octets_out
The total number of bytes (not including FCSs) transmitted to adjacent link stations.

protocol_errs
The number of times that Communications Server for Linux has deactivated an LS using this port because a frame received from the adjacent link station contained a protocol error.

activity_to_s
The number of times that Communications Server for Linux has deactivated an LS using this port because there was no activity on the link.

rnrlimit_s
The number of times that Communications Server for Linux has deactivated an LS using this port because the remote busy timer expired.

retries_exps
The number of times that Communications Server for Linux has deactivated an LS using this port because a retry sequence has been exhausted.

retransmits_in
The total number of retransmitted I-frames received from adjacent link stations.

retransmits_out
The total number of retransmissions of I-frames to adjacent link stations.

Returned Parameters: SDLC Port Operational Information

Parameter name	Type
index	decimal
role	constant
type	constant
topology	constant
activto	decimal
pause	decimal
slow_poll_method	constant
slow_poll_timer	decimal

If the command executes successfully, the following parameters are returned:

index The index value used internally by Communications Server for Linux to identify the port.

role The link role of the port. Possible values are:

PRIMARY
Port is the primary link.

SECONDARY
Port is the secondary link.

NEGOTIABLE
Port role is negotiable.

type Specifies whether the port is operating as though connected to a leased or switched line. Possible values are:

LEASED Port is operating as though connected to a leased line.

SWITCHED
Port is operating as though connected a switched line.

topology
Specifies whether the port can operate in a multipoint topology. Possible values are:

POINT_TO_POINT
Port can operate only as point-to-point.

MULTIPOINT
Port can operate as multipoint.

activto The length of time, in hundredths of a second, that the port allows a switched line to remain inactive (no I-frames being transferred) before disconnecting. A value of `0` indicates no timeout; the line remains connected regardless of inactivity. This parameter is defined only for a switched link; its value is undefined for a leased link.

pause The length of time that the primary station waits between successive cycles of polling secondary stations. This parameter is defined only if the LS role is `PRIMARY`; its value is undefined if the LS role is `SECONDARY`.

slow_poll_method
The method used for periodically polling failed secondary link stations. This parameter is set to `POLLPAUSE`.

slow_poll_timer
The timeout between polls for failed secondary link stations. This parameter applies only if the port is `PRIMARY` and operating in a multipoint topology; its value is undefined otherwise.

Returned Parameters: Token Ring / Ethernet LS Statistics

Parameter name	Type	Length
local_mac	hex array	6
local_sap	hex number	
remote_mac	hex array	6
remote_sap	hex number	
rif	(see notes)	8
state	decimal	
max_btu_size	decimal	
send_window	decimal	
receive_window	decimal	
t1_expiry_count	decimal	
t2_expiry_count	decimal	
remote_busy	decimal	
local_busy	decimal	
i_frames_sent	decimal	
i_bytes_sent	decimal	
i_frames_rcvd	decimal	
i_bytes_rcvd	decimal	
i_frames_rjctd	decimal	
i_bytes_rjctd	decimal	
i_frames_rexmit	decimal	
i_bytes_rexmit	decimal	
rej_frames_sent	decimal	
rej_frames_rcvd	decimal	
xid_frames_sent	decimal	
xid_frames_rcvd	decimal	

```
ack_timeout             decimal
p_bit_timeout           decimal
t2_timeout              decimal
rej_timeout             decimal
busy_state_timeout      decimal
idle_timeout            decimal
max_retry               decimal
```

If the command executes successfully, the following parameters are returned:

local_mac
The MAC address of the local link station.

local_sap
The SAP address of the local link station.

remote_mac
The MAC address of the remote link station.

remote_sap
The SAP address of the remote link station.

rif Routing Information Field data. This parameter is used only for Token Ring; it is reserved for other DLC types.

The data is returned as an array of pairs of decimal values, for example <321/4, 1234/8, 2345/12>. The first value in each pair specifies the ring number and the second value specifies the bridge number.

state An internal value indicating the processing state of the LS software (for use by support personnel).

max_btu_size
The maximum BTU size determined during LS activation. This value includes the length of the TH and RH (total 9 bytes), as well as the RU.

send_window
The number of I-frames the local station can send to the adjacent station before it must wait for a response.

receive_window
The number of I-frames the adjacent station can send to the local station before it must wait for a response.

t1_expiry_count
The number of times the adjacent station failed to respond within the *t1_timeout* (acknowledgment timeout) period.

t2_expiry_count
The number of times the *t2_timeout* period expired before a frame that could carry the required reply bits was queued.

remote_busy
The number of times the local station entered remote busy state because of an RNR frame from the adjacent station.

local_busy
The number of times the local station sent an RNR frame to the adjacent station on entering local busy state.

i_frames_sent
The number of I-frames sent.

i_bytes_sent
The number of data bytes in the I-frames sent.

i_frames_rcvd
The number of I-frames received.

i_bytes_rcvd
The number of data bytes in the I-frames received.

i_frames_rjctd
The number of I-frames rejected.

i_bytes_rjctd
The number of data bytes in the I-frames rejected.

i_frames_rexmit
The number of I-frames retransmitted.

i_bytes_rexmit
The number of data bytes in the I-frames retransmitted.

rej_frames_sent
The number of REJ frames sent to request retransmission of one or more I-frames.

rej_frames_rcvd
The number of REJ frames received requesting retransmission of one or more I-frames.

xid_frames_sent
The number of XID frames sent.

xid_frames_rcvd
The number of XID frames received.

ack_timeout
An acknowledgment timeout—the time in milliseconds within which a response must be received for any I-frames sent to the adjacent link station.

p_bit_timeout
A poll bit timeout—the time in milliseconds within which a response must be received for any frames sent to the adjacent link station with the POLL bit set.

t2_timeout
The t2 timeout—the maximum time in milliseconds that the local station can wait before it must send a response to a received I-frame. A longer timeout allows the local station to respond to more than one I-frame with a single RR, and so reduces acknowledgment traffic.

rej_timeout
The reject timeout—the time in seconds within which a response must be received for a REJ frame sent to the adjacent link station.

busy_state_timeout
The busy state timeout—the time in seconds that the local station waits for indication from the adjacent link station that a busy state (RNR) has cleared.

idle_timeout
The idle timeout is used to detect a completely inactive line. The line is considered idle when nothing has been received in this time. The timer is specified in seconds.

max_retry
The maximum number of times that the local station will retry when waiting for a response or for a busy state to clear.

Returned Parameters: Token Ring or Ethernet Port Statistics

Parameter name	Type	Length
time_secs	decimal	
time_ms	decimal	
mac_addr	hex array	6
max_btu_size	decimal	
ls_count	decimal	
ui_frames_sent	decimal	
ui_frames_rcvd	decimal	
adapter_number	decimal	
line_error	decimal	
internal_error	decimal	
burst_error	decimal	
ari_fci_error	decimal	
end_delim	decimal	
lost_frame	decimal	
rcv_cngstn	decimal	
frm_cpy_err	decimal	
freq_err	decimal	
token_err	decimal	
crc_err	decimal	
xmit_err	decimal	
collision_err	decimal	

If the command executes successfully, the following parameters are returned:

time_secs
The time, in seconds, from when the SNA software was started to when the LLC2 component received the port activation request.

time_ms
The time, in milliseconds, from when the SNA software was started to when the LLC2 component received the port activation request.

mac_addr
The MAC address of the port, determined during port activation.

max_btu_size
The maximum BTU size, determined during port activation. This value includes the length of the TH and RH (total 9 bytes), as well as the RU.

ls_count
The number of link stations currently using the port. This number includes stations for which XIDs have been sent but SABME has not yet been sent.

ui_frames_sent
The total number of unnumbered TEST and XID frames issued on this port.

ui_frames_rcvd
The total number of unnumbered TEST and XID frames received on this port.

line_error
The total number of line errors.

internal_error
The total number of internal errors.

burst_error
The total number of burst errors.

ari_fci_error
The total number of address recognized / frame copied bits errors.

end_delim
The total number of frame delimiter errors.

lost_frame
The total number of lost frame errors.

rcv_cngstn
The total number of receiver congestion errors.

frm_cpy_err
The total number of frame copied errors.

freq_err
The total number of frequency errors.

token_err
The total number of token errors.

crc_err The total number of CRC (cyclic redundancy check) errors.

xmit_err
The total number of transmit errors.

collision_err
The total number of collision errors.

Returned Parameters: Enterprise Extender

Parameter name	Type	Length
udp_low_out	decimal	
udp_med_out	decimal	
udp_high_out	decimal	
udp_network_out	decimal	
udp_llc_out	decimal	

If the command executes successfully, the following parameters are returned:

udp_low_out
The number of UDP datagrams sent that contained low priority APPN data.

udp_med_out
The number of UDP datagrams sent that contained medium priority APPN data.

udp_high_out
The number of UDP datagrams sent that contained high priority APPN data.

udp_network_out
The number of UDP datagrams sent that contained network priority APPN data.

udp_llc_out
The number of UDP datagrams sent that contained LLC commands.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_LINK_NAME
The supplied *name* parameter was not a valid LS name.

INVALID_PORT_NAME
The supplied *name* parameter was not a valid port name.

INVALID_STATS_TYPE
The *stats_type* parameter was not set to a valid value.

INVALID_TABLE_TYPE
The *table_type* parameter was not set to a valid value.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

LINK_DEACTIVATED
The specified link is not currently active.

PORT_DEACTIVATED
The specified port is not currently active.

Function Not Supported

If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameter:

primary_rc

FUNCTION_NOT_SUPPORTED
The DLC type does not support returning statistics information.

secondary_rc
(This parameter is not used.)

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_tn3270_access_def

The **query_tn3270_access_def** command returns information that was supplied on a **define_tn3270_access** command about TN3270 clients that can use the TN server feature of Communications Server for Linux to access a host for 3270 emulation using TN3270 Server. (To return information about users accessing the host using TN Redirector, use **query_tn_redirect_def**.).

The **query_tn3270_access_def** command can return summary or detailed information about a single client or multiple clients, depending on the options used.

Supplied Parameters

Parameter name [query_tn3270_access_def]	Type	Length	Default
num_entries	decimal		1
list_options	constant		SUMMARY + LIST_INCLUSIVE
default_record	constant		NO
client_address	character	256	(null string)
port_number	decimal		(none specified)

Supplied parameters are:

num_entries
: Maximum number of clients for which data should be returned. If detailed information about client sessions is being returned, this number includes partial entries (entries with a specified client address; the returned data does not include the client definition or the client's first session). You can specify 1 to return data for a specific client, a number greater than 1 to return data for multiple clients, or 0 to return data for all clients.

list_options
: The level of information required for each client and the position in the list of clients from which Communications Server for Linux begins to return data.

  Specify the level of information required with one of the following values:

  SUMMARY
  : Summary information only

  DETAIL Detailed information

  Use a + character to combine this value with one of the following values:

  FIRST_IN_LIST
  : Start at the first session for the first client in the list

  LIST_INCLUSIVE
  : Start at the session specified by the supplied *client_address* and *port_number* parameters, or start at the first session for the specified client address if no port number is specified

  LIST_FROM_NEXT
  : Start at the session immediately following the session specified by the *client_address* and *port_number* parameters, or start at the first session for the specified client address if no port number is specified

default_record
: Specifies whether the requested entry (or the entry to be used as an index

into the list) is the default record. This parameter is ignored if *list_options* is set to FIRST_IN_LIST. Possible values are:

YES The requested entry is the default record. Use this parameter to query the default access record used by a TN3270 client not explicitly identified by a TCP/IP address. Do not specify the *client_address* parameter.

NO The requested entry is not the default record. Use this parameter to query the access record for the client specified by the *client_address* parameter.

client_address
The TCP/IP address of the TN3270 client for whom information is required, or the name to be used as an index into the list of clients. This parameter is ignored if *list_options* is set to FIRST_IN_LIST. This address can be specified as any of the following.

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

port_number
This parameter is ignored if *list_options* is set to SUMMARY.

If *list_options* is set to DETAILED, to return information starting with or immediately following a specific session entry, specify a value for the *client_address* parameter and set this parameter to the TCP/IP port number defined for that session. To return information starting with the first session entry, specify a value for the *client_address* parameter and do not specify a value for this parameter.

Returned Parameters: Summary Information

Parameter name	Type	Length
default_record	constant	
client_address	character	256
address_format	constant	

If the command executes successfully and you specified SUMMARY as the *list_options* parameter value, Communications Server for Linux returns the following parameters:

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

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

NO This entry is a TN3270 record for the specified client address.

client_address
The TCP/IP address of the TN3270 client. This can be any of the following; the *address_format* parameter indicates whether it is an IP address or a name.

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

- An alias (such as newbox).

address_format
Specifies the format of the *client_address* parameter. Possible values are:

IP_ADDRESS
IP address (either IPv4 or IPv6)

FULLY_QUALIFIED_NAME
Alias or fully qualified name

If you specified SUMMARY as the *list_options* parameter value, only summary information about TN3270 clients is returned; no information about sessions on those clients is returned. To obtain information about sessions, set the *list_options* parameter to DETAIL.

Returned Parameters: Detailed Information

If the command executes successfully and you specified DETAIL as the *list_options* parameter value, Communications Server for Linux returns a sequence of client entries, identified by the *client_address* parameter (unless the entry is a default record as identified by the *default_record* parameter set to YES), with each client entry followed by session entries for that client. Each session entry is identified by the *port_number* parameter.

The following parameters are returned for each TN3270 client:

```
Parameter name              Type        Length
default_record              constant
client_address              character   256
description                 character   31
address_format              constant
num_sessions                decimal
```

The following parameters are returned for each session on the TN3270 client:

```
description                 character   31
port_number                 decimal
lu_name                     character   8
printer_lu_name             character   8
tn3270_support              constant
allow_specific_lu           constant
ssl_enabled                 constant
security_level              constant
cert_key_label              character   80
allow_ssl_timeout_to_nonssl constant
```

The following parameters are returned for each client entry:

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

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

NO This entry is a TN3270 record for a specified client.

client_address
The TCP/IP address of the TN3270 client. This can be any of the following; the *address_format* parameter indicates whether it is an IP address or a name.

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

- A name (such as `newbox.this.co.uk`).
- An alias (such as `newbox`).

description
An optional string describing the client.

address_format
Specifies the format of the *client_address* parameter. Possible values are:

IP_ADDRESS
IP address (either IPv4 or IPv6)

FULLY_QUALIFIED_NAME
Alias or fully qualified name

num_sessions
Indicates the number of subrecords (session entries) for the client.

Additional parameters are returned for each session entry for a given client entry (unless the entry is a default record). For each session defined for the specified client (defined by its TCP/IP address), the following parameters are returned:

description
An optional string describing the session.

port_number
The number of the TCP/IP port that the TN3270 emulator uses to access the TN server node.

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

printer_lu_name
Name of the printer LU or LU pool that this session uses for connections requesting a generic printer LU.

tn3270_support
Specifies the level of TN3270 support. Possible values are:

TN3270 Specifies that TN3270E protocols are disabled.

TN3270E
Specifies that TN3270E protocols are enabled.

TN3270 and TN3287 protocols are always enabled.

allow_specific_lu
Indicates whether access to specific LUs is allowed. Possible values are:

YES Access to specific LUs is allowed. Clients are allowed to request access to a specific LU or LU pool; clients do not have to use the LU or LU pool chosen by TN server.

NO Access to specific LUs is not allowed.

ssl_enabled
Indicates whether this session uses Secure Sockets Layer (SSL) to access the server.

SSL support is available only if you have installed the additional software required to support SSL on the server. You can check this by using the **query_node_limits** command and checking the value of the *ssl_support* parameter.

Possible values are:

NO This session does not use SSL.

YES This session uses SSL.

YES_WITH_CLI_AUTH
This session uses SSL, and the TN Server requires it to use client authentication. The client must send a valid certificate (information identifying it as a valid client authorized to use the TN Server).

security_level
Indicates the SSL security level required for this session. The session will use the highest security level that both client and server can support; if the client cannot support the requested level of security or higher, the session will not be started.

If the *ssl_enabled* parameter is set to NO, this parameter is not used.

Possible values are:

SSL_AUTHENTICATE_MIN
Certificates must be exchanged; encryption is not required (but can be used if the client requests it).

SSL_AUTHENTICATE_ONLY
Certificates must be exchanged, but encryption will not be used. This option is typically used to avoid the overhead of encryption when the client is connecting across a secure intranet.

SSL_40_BIT_MIN
Use at least 40-bit encryption.

SSL_56_BIT_MIN
Use at least 56-bit encryption.

SSL_128_BIT_MIN
Use at least 128-bit encryption.

SSL_168_BIT_MIN
Use at least 168-bit encryption.

Note: Using encryption requires additional software to be installed with Communications Server for Linux; see *Communications Server for Linux Quick Beginnings* for more information. Depending on your location, you may not be able to use all the encryption levels listed because the software required to support them is not available in your country.

cert_key_label
The label identifying a certificate and key pair for use with SSL on this session. This must match a label specified when the SSL keyring database was set up; see *Communications Server for Linux Quick Beginnings* for more information.

If this parameter is not shown, this indicates that the session uses the default SSL certificate and key pair, specified when the SSL keyring database was set up.

allow_ssl_timeout_to_nonssl
This parameter does not apply if *ssl_enabled* is set to NO. Indicates whether non-SSL TN3270 clients can access the server using this session record even though it is configured to use SSL. Possible values are:

YES TN3270 clients not using SSL can access the server. There will be a

5–second delay on startup while the server waits for SSL negotiation to begin; after this, the server will assume that the client is not using SSL and revert to normal TN3270 communications.

NO Only TN3270 clients using SSL can access the server.

Note: This option is provided for migration purposes: if you have large numbers of clients that use the same port, and are migrating them from non-SSL to SSL configuration, you can set up the configuration to accept both SSL and non-SSL connections on the same port while the migration is in progress.

Allowing non-SSL clients to use SSL resources may be a security exposure, so this option is not intended for long-term use. You should set this parameter to `YES` only for brief periods while migration is in progress, and then set it to `NO` when migration is complete.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
`PARAMETER_CHECK`

secondary_rc
Possible values are:

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 client.

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 client.

State Check

No specific state errors, other than those listed in Appendix A, “Common Return Codes from snaadmin Commands,” on page 589, occur for this command.

Other Conditions

Appendix A, “Common Return Codes from snaadmin Commands,” on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_tn3270_association

The **query_tn3270_association** command returns information about associations between display LUs and printer LUs, as defined on **define_tn3270_association**. Associations are queried by display LU name and are returned in order of display LU name. This command can be used to obtain information about a specific association or about multiple associations, depending on the options used.

Supplied Parameters

Parameter	Type	Length	Default
[query_tn3270_association]			
num_entries	decimal		1
list_options	constant		LIST_INCLUSIVE
display_lu_name	character	8	(null string)

Supplied parameters are:

num_entries
: Maximum number of associations for which data should be returned. You can specify 1 to return data for a specific association, a number greater than 1 to return data for multiple associations, or 0 to return data for all associations.

list_options
: The position in the list of associations from which Communications Server for Linux begins to return data.

  Possible values are:

  FIRST_IN_LIST
  : Start at the first entry in the list

  LIST_INCLUSIVE
  : Start at the entry specified by the *display_lu_name* parameter

  LIST_FROM_NEXT
  : Start at the entry immediately following the entry specified by the *display_lu_name* parameter

display_lu_name
: Name of the display LU for which association information is required, or the name to be used as an index into the list of associations. The display LU name is an 8-byte character string. This parameter is ignored if *list_options* is set to FIRST_IN_LIST.

Returned Parameters

Parameter	Type	Length
display_lu_name	character	8
printer_lu_name	character	8
description	character	31

If the command executes successfully, Communications Server for Linux returns the following parameters:

display_lu_name
: Name of the display LU associated with the printer LU specified by the *printer_lu_name* parameter.

printer_lu_name
: Name of the printer LU associated with the display LU specified by the *display_lu_name* parameter.

description
An optional text string describing the association.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

INVALID_LU_NAME
The *list_options* parameter was set to LIST_INCLUSIVE, but the display LU specified in the *display_lu_name* parameter did not match any existing association.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

query_tn3270_defaults

The **query_tn3270_defaults** command returns information about TN3270 parameters used on all client sessions, as defined on **define_tn3270_defaults**.

If you are using Secure Sockets Layer (SSL) client authentication, and checking clients against a certificate revocation list on an external LDAP server, use the **query_tn3270_ssl_ldap** command to return details of how to access this server.

Supplied Parameters

```
[query_tn3270_defaults]
```

No parameters are supplied for this command.

Returned Parameters

Parameter	Type	Length
force_responses	constant	
keepalive_method	constant	
keepalive_interval	decimal	

If the command executes successfully, Communications Server for Linux returns the following parameters:

force_responses
Controls client responses on printer sessions. Possible values are:

YES Request definite responses.

NO Request responses matching SNA traffic.

keepalive_method
Method for sending keep-alive messages. Keep-alive messages are messages sent to TN3270 clients when there is no other activity on the connection, to keep the TCP/IP connections to the clients active; this ensures that failed connections and clients can be detected. If there is no traffic at all on a TCP/IP connection, failure of the connection or of the client may never be detected, which wastes TN server resources and prevents LUs from being used for other sessions.

Possible values are:

NONE Do not send keep-alive messages.

NOP Send Telnet NOP messages.

TM Send Telnet DO TIMING-MARK messages.

keepalive_interval
Interval (in seconds) between consecutive keep-alive messages. The interval should be long enough to minimize network traffic, especially if there are typically many idle client connections. The shorter the keep-alive interval, the quicker failures are detected, but the more network traffic is generated. If the keep-alive interval is too short and there are many clients, this traffic can be significant.

Because of the way TCP/IP operates, the keepalive interval that you configure is not the exact time that it will take for the server to recognize that a client has disappeared. The configured interval is the minimum time in which a client could be timed out. The maximum is approximately twice the configured timeout, plus a few extra minutes (the exact number depends on how TCP/IP is configured).

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_tn3270_express_logon

The **query_tn3270_express_logon** command returns information about the TN3270 Express Logon feature. This feature means that TN3270 client users who connect to Communications Server for Linux TN Server or TN Redirector using the Secure Sockets Layer (SSL) client authentication feature do not need to supply the user ID and password normally used for TN3270 security. Instead, their security certificate

is checked against a Digital Certificate Access Server (DCAS) at the host, which supplies the required user ID and password.

Supplied Parameters

```
[query_tn3270_express_logon]
```

No parameters are supplied for this command.

Returned Parameters

Parameter	Type	Length
dcas_server	character	255
dcas_port	decimal	
enabled	constant	

If the command executes successfully, Communications Server for Linux returns the following parameters:

dcas_server
The TCP/IP address of the host DCAS server that handles Express Logon authorization. This can be specified as any of the following.

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

dcas_port
The TCP/IP port number used to access the DCAS server.

enabled Specifies whether the TN3270 Express Logon function is enabled. Possible values are:

YES The function is enabled, so TN3270 clients can access the host without needing to specify a user ID and password.

NO The function is not enabled, so TN3270 clients must specify a user ID and password.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_tn3270_ssl_ldap

The **query_tn3270_ssl_ldap** command returns information about how to access a certificate revocation list for use with the Secure Sockets Layer (SSL) client authentication feature. This information was specified using the **define_tn3270_ssl_ldap** command.

Supplied Parameters

```
[query_tn3270_ssl_ldap]
```

No parameters are supplied for this command.

Returned Parameters

Parameter	Type	Length
auth_type	constant	
ldap_addr	character	255
ldap_port	decimal	
ldap_user	character	1024
ldap_password	character	128

If the command executes successfully, Communications Server for Linux returns the following parameters:

auth_type
Specifies the type of authorization checking performed by the TN Server or TN Redirector. Possible values are:

LOCAL_ONLY
The server checks client certificates locally, but does not use an external certificate revocation list. The parameters *ldap_addr*—*ldap_password* are not used.

LOCAL_X500
The server checks certificates locally, and also checks against an external certificate revocation list. The remaining returned parameters specify the location of this list.

ldap_addr
The TCP/IP address of the LDAP server that holds the certificate revocation list. This can be specified as any of the following.

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

ldap_port
The TCP/IP port number used to access the LDAP server.

ldap_user
The user name used to access the certificate revocation list on the LDAP server.

ldap_password
The password used to access the certificate revocation list on the LDAP server.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

query_tn_redirect_def

The **query_tn_redirect_def** command returns information that was supplied on a **define_tn_redirect** command about Telnet clients that can use the TN Redirector feature of Communications Server for Linux to access a host. The command can return summary or detailed information about a single client or multiple clients, depending on the options used.

Supplied Parameters

Parameter name	Type	Length	Default
[query_tn_redirect_def]			
num_entries	decimal		1
list_options	constant		LIST_INCLUSIVE
default_record	constant		NO
client_address	character	256	(null string)
client_port	decimal		(none specified)

Supplied parameters are:

num_entries
: Maximum number of clients for which data should be returned. You can specify 1 to return data for a specific client, a number greater than 1 to return data for multiple clients, or 0 to return data for all clients.

list_options
: The position in the list of clients from which Communications Server for Linux begins to return data. Specify one of the following values:

  FIRST_IN_LIST
  : Start at the first client in the list

  LIST_INCLUSIVE
  : Start at the entry specified by the supplied *client_address* and *port_number* parameters

  LIST_FROM_NEXT
  : Start at the entry immediately following the entry specified by the *client_address* and *port_number* parameters

default_record
: Specifies whether the requested entry (or the entry to be used as an index

into the list) is the default record. This parameter is ignored if *list_options* is set to FIRST_IN_LIST. Possible values are:

YES The requested entry is the default record. Use this option to query the default access record used by a Telnet client not explicitly identified by a TN Redirector access record. Do not specify the *client_address* parameter.

NO The requested entry is not the default record. Use this option to query the access record for the client specified by the *client_address* parameter.

client_address
The TCP/IP address of the Telnet client for whom information is required, or the client to be used as an index into the list of clients. This parameter is ignored if *list_options* is set to FIRST_IN_LIST. The address can be specified as any of the following.

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

client_port
The TCP/IP port number used by the client. This parameter is ignored if *list_options* is set to FIRST_IN_LIST.

Returned Parameters

If the command executes successfully, Communications Server for Linux returns the following parameters:

Parameter name	Type	Length
default_record	constant	
client_address	character	256
client_port	decimal	
cli_conn_ssl_enabled	constant	
cli_conn_security_level	constant	
cli_conn_cert_key_label	character	80
host_address	character	255
host_port	decimal	
serv_conn_ssl_enabled	constant	
serv_conn_security_level	constant	
serv_conn_cert_key_label	character	80
description	character	31

The following parameters are returned for each client entry:

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

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

NO This entry is a TN Redirector record for a specified client.

client_address
The TCP/IP address of the Telnet client. This can be any of the following.

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

- An alias (such as `newbox`).

client_port
The number of the TCP/IP port that the Telnet client uses to access the TN server node.

cli_conn_ssl_enabled
Indicates whether the client uses Secure Sockets Layer (SSL) to access TN Redirector. Possible values are:

NO The client does not use SSL.

YES The client uses SSL.

YES_WITH_CLI_AUTH
The client uses SSL, and the TN Redirector requires it to use client authentication. The client must send a valid certificate (information identifying it as a valid client authorized to use the TN Redirector).

As well as checking that the certificate is valid, the TN Redirector may also need to check the certificate against a certificate revocation list on an external LDAP server, to ensure that the user's authorization has not been revoked. In this case, you also need to use **define_tn3270_ssl_ldap** to specify how to access this server.

cli_conn_security_level
Indicates the SSL security level required for the client connection on this session. The session will use the highest security level that both client and server can support; if the client cannot support the requested level of security or higher, the session will not be started.

If the *cli_conn_ssl_enabled* parameter is set to `NO`, this parameter is not used.

Possible values are:

SSL_AUTHENTICATE_MIN
Certificates must be exchanged; encryption is not required (but can be used if the client requests it).

SSL_AUTHENTICATE_ONLY
Certificates must be exchanged, but encryption will not be used. This option is typically used to avoid the overhead of encryption when the client is connecting across a secure intranet.

SSL_40_BIT_MIN
Use at least 40-bit encryption.

SSL_56_BIT_MIN
Use at least 56-bit encryption.

SSL_128_BIT_MIN
Use at least 128-bit encryption.

SSL_168_BIT_MIN
Use at least 168-bit encryption.

Note: Using encryption requires additional software to be installed with Communications Server for Linux; see *Communications Server for Linux Quick Beginnings* for more information. Depending on your location, you may not be able to use all the encryption levels listed because the software required to support them is not available in your country.

cli_conn_cert_key_label
The label identifying a certificate and key pair for use with SSL on the client session. This must match a label specified when the SSL keyring database was set up; see *Communications Server for Linux Quick Beginnings* for more information.

If the *cli_conn_ssl_enabled* parameter is set to `NO`, this parameter is not used.

If this parameter is not specified, this indicates that the session uses the default SSL certificate and key pair, specified when the SSL keyring database was set up.

host_address
The TCP/IP address of the host computer with which the client communicates. This can be specified as any of the following.

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

host_port
The number of the TCP/IP port that the TN Redirector node uses to access the host.

serv_conn_ssl_enabled
Indicates whether the TN Redirector uses Secure Sockets Layer (SSL) to access the host on behalf of this client. Possible values are:

NO The host does not use SSL.

YES The host uses SSL.

serv_conn_security_level
Indicates the SSL security level required for the host connection on this session. The session will use the highest security level that both host and server can support; if the host cannot support the requested level of security or higher, the session will not be started.

If the *serv_conn_ssl_enabled* parameter is set to `NO`, this parameter is not used.

Possible values are:

SSL_AUTHENTICATE_MIN
Certificates must be exchanged; encryption is not required (but can be used if the host requests it).

SSL_AUTHENTICATE_ONLY
Certificates must be exchanged, but encryption will not be used. This option is typically used to avoid the overhead of encryption when the host connection is across a secure intranet.

SSL_40_BIT_MIN
Use at least 40-bit encryption.

SSL_56_BIT_MIN
Use at least 56-bit encryption.

SSL_128_BIT_MIN
Use at least 128-bit encryption.

SSL_168_BIT_MIN
Use at least 168-bit encryption.

Note: Using encryption requires additional software to be installed with Communications Server for Linux; see *Communications Server for Linux Quick Beginnings* for more information. Depending on your location, you may not be able to use all the encryption levels listed because the software required to support them is not available in your country.

serv_conn_cert_key_label
The label identifying a certificate and key pair for use with SSL on the host session. This must match a label specified when the SSL keyring database was set up; see *Communications Server for Linux Quick Beginnings* for more information.

If the *serv_conn_ssl_enabled* parameter is set to NO, this parameter is not used.

If this parameter is not specified, this indicates that the session uses the default SSL certificate and key pair, specified when the SSL keyring database was set up.

description
An optional string describing the client.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_CLIENT_ADDRESS
The *list_options* parameter was set to LIST_INCLUSIVE, but the supplied addressing information did not match the address of any defined Telnet client.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_tn_server_trace

The **query_tn_server_trace** command returns information about the current tracing options for the Communications Server for Linux TN server feature.

This command must be issued to a running node.

Supplied Parameters

```
[query_tn_server_trace]
```

No parameters are supplied for this command.

Returned Parameters

```
Parameter name          Type
trace_flags             constant
```

If the command executes successfully, Communications Server for Linux returns the following parameters:

trace_flags
The types of tracing currently active.

If no tracing is active, or if tracing is active for all types of messages, one of the following values is returned:

NONE No tracing is active.

ALL Tracing of all types of messages is active.

If tracing is used on specific message types, Communications Server for Linux returns one or more of the following values are returned (combined using a + character):

TCP Messages between TN server and TN3270 clients are traced.

FMAPI Internal control messages and messages between TN server and TN3270 clients (in internal format) are traced.

CFG Messages relating to the configuration of TN server are traced.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_tp

The **query_tp** command returns information about transaction programs (TPs) currently being used by a local LU. It can be used to obtain information about a specific TP or about multiple TPs, depending on the options used. This command returns information about current usage of the TPs, not about their definition; use **query_tp_definition** to obtain the definition of the TPs.

This command must be issued to a running node.

Supplied Parameters

Parameter name	Type	Length	Default
[query_tp]			
num_entries	decimal		1
list_options	constant		LIST_INCLUSIVE
lu_name	character	8	(null string)
lu_alias	character	8	(null string)
tp_name	character	64	(null string)

Supplied parameters are:

num_entries
: Maximum number of TPs for which data should be returned. You can specify 1 to return data for a specific TP, a number greater than 1 to return data for multiple TPs, or 0 to return data for all TPs.

list_options
: The position in the list of TPs from which Communications Server for Linux begins to return data.

  Possible values are:

  FIRST_IN_LIST
  : Start at the first entry in the list

  LIST_INCLUSIVE
  : Start at the entry specified by the combination of *lu_name*, *lu_alias*, and *tp_name* parameters

  LIST_FROM_NEXT
  : Start at the entry immediately following the entry specified by the combination of *lu_name*, *lu_alias*, and *tp_name* parameters

lu_name
: LU name of the local LU. This name is a type-A character string. To indicate that the LU is identified by its LU alias instead of its LU name, do not specify this parameter. To specify the LU associated with the local CP (the default LU), do not specify either *lu_name* or *lu_alias*.

lu_alias
: Locally defined LU alias. This parameter is used only if *lu_name* is not specified. To specify the LU associated with the local CP (the default LU), do not specify either *lu_name* or *lu_alias*.

tp_name
: TP name for which information is required. This value is ignored if *list_options* is set to FIRST_IN_LIST.

Returned Parameters

Parameter name	Type	Length
tp_name	character	64
description	character	31
instance_limit	decimal	
instance_count	decimal	
locally_started_count	decimal	
remotely_started_count	decimal	

If the command executes successfully, Communications Server for Linux returns the following parameters:

tp_name
TP name.

description
A text string describing the TP, as specified in the definition of the TP.

instance_limit
Maximum number of simultaneously active instances of the specified TP.

instance_count
Number of instances of the specified TP that are currently active.

locally_started_count
Number of instances of the specified TP which have been started locally (by the TP issuing a TP_STARTED verb).

remotely_started_count
Number of instances of the specified TP that have been started remotely (by receiving an Attach request).

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_LU_ALIAS
The *lu_alias* parameter value was not valid.

INVALID_LU_NAME
The *lu_name* parameter value was not valid.

INVALID_TP_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *tp_name* parameter value was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_tp_definition

The **query_tp_definition** command returns information about transaction programs (TPs) defined on a Communications Server for Linux system. It can be used to obtain information about a specific TP or about multiple TPs, depending on the options used. This command returns information about the definition of the TPs, not about their current usage; use **query_tp** to obtain usage information.

Supplied Parameters

Parameter name	Type	Length	Default
[query_tp_definition]			
num_entries	decimal		1
list_options	constant		SUMMARY + LIST_INCLUSIVE
tp_name	character	64	(null string)

Supplied parameters are:

num_entries
: Maximum number of TPs for which data should be returned. You can specify 1 to return data for a specific TP, a number greater than 1 to return data for multiple TPs, or 0 to return data for all TPs.

list_options
: The level of information required for each entry and the position in the list of TPs from which Communications Server for Linux begins to return data.

  Specify the level of information required with one of the following values:

  SUMMARY
  : Summary information only

  DETAIL Detailed information

  Use a + character to combine this value with one of the following values:

  FIRST_IN_LIST
  : Start at the first entry in the list

  LIST_INCLUSIVE
  : Start at the entry specified by the *tp_name* parameter

  LIST_FROM_NEXT
  : Start at the entry immediately following the entry specified by the *tp_name* parameter

tp_name
: TP name for which information is required or the name to be used as an index into the list of TPs. This value is ignored if *list_options* is set to FIRST_IN_LIST.

Returned Parameters: Summary Information

Parameter name	Type	Length
tp_name	character	64
description	character	31

If the command executes successfully and you specified SUMMARY as the *list_options* parameter value, Communications Server for Linux returns the following parameters:

tp_name
TP name.

description
A text string describing the TP, as specified in the definition of the TP.

Returned Parameters: Detailed Information

Parameter name	Type	Length
tp_name	character	64
description	character	31
list_name	character	14
conv_type	constant	
security_rqd	constant	
sync_level	constant	
enabled	constant	
pip_allowed	constant	
tp_instance_limit	decimal	
incoming_alloc_timeout	decimal	

If the command executes successfully and you specified DETAIL as the *list_options* parameter value, Communications Server for Linux returns the following parameters:

tp_name
TP name.

description
A text string describing the TP, as specified in the definition of the TP.

list_name **through** *incoming_alloc_timeout*
For information about these parameters, see "define_tp" on page 203.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_TP_NAME
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied name, but the *tp_name* parameter value was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_tp_load_info

The **query_tp_load_info** command returns information about TP load information entires..

Supplied Parameters

Parameter name	Type	Length	Default
[query_tp_load_info]			
num_entries	decimal	1	
list_options	constant		LIST_INCLUSIVE
tp_name	character	64	(null string)
lualias	character	8	(null string)

Supplied parameters are:

num_entries
Maximum number of extra data control blocks for which data should be returned. You can specify 1 to return data for a specific data control block, a number greater than 1 to return data for multiple data control blocks, or 0 (zero) to return data for the maximum number of data control blocks that can be accommodated in the supplied data buffer.

list_options
The position in the list from which Communications Server for Linux begins to return data.

Possible values are:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the combination of the *tp_name* and *lualias* parameters.

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the combination of the *tp_name* and *lualias* parameters.

tp_name
The name of the TP to query. This name is a 64-byte string. This value is ignored if *list_options* is set to FIRST_IN_LIST. If no *tp_name* is specified, the command returns information about all TPs.

lualias The LU alias to query. This alias is an 8-byte string. If no *lualias* is specified, then the command returns information about all LUs.

This parameter can be used only if the TP is an APPC application; it must not be specified if the TP is a CPI-C application.

Returned Parameters

Parameter name	Type	Length
tp_name	character	64
lualias	character	8
description	character	31
path	character	255

```
arguments                 character    255
type                      constant
timeout                   decimal
userid                    character    64
group                     character    64
stdin                     character    255
stdout                    character    255
stderr                    character    255
env                       character    255
```

If the command executes successfully, Communications Server for Linux returns the following parameters:

tp_name
The TP name of the TP load info entry.

lualias The LU alias of the TP load info entry.

This parameter is used only if the TP is an APPC application; it is not used if the TP is a CPI-C application.

description
Optional text string describing the TP load info.

path The full path name of the TP executable.

arguments
Command-line arguments required by the TP. These arguments are separated with spaces.

type Specifies the TP type. Possible values are:

QUEUED The TP is a queued TP.

QUEUED-BROADCAST
The TP is a broadcast queued TP.

NON-QUEUED
The TP is a nonqueued TP.

timeout
Timeout in seconds after the TP is loaded. The value -1 indicates an infinite timeout.

userid User ID required to access and run the TP.

group Group ID required to access and run the TP.

stdin Full path name of standard input file or device.

stdout Full path name of standard output file or device.

stderr Full path name of standard error file or device.

env Environment variables required by the TP in the form *VARIABLE* = *VALUE*.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

INVALID_TP_NAME
The *tp_name* parameter specified did not match the name of a defined TP.

State Check
No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions
Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_trace_file

The **query_trace_file** command returns information about the files that Communications Server for Linux uses to record trace data.

This command may be issued to a running node or (for client/server trace files only) to a Remote API Client on AIX or Linux. To issue the command to a client computer, use the **snaadmin** program on the client computer without specifying a node name.

On Windows clients, tracing is controlled by options in the Windows Registry. For more information, refer to *Communications Server for Linux Diagnostics Guide*.

Supplied Parameters

Parameter name	Type	Length	Default
[query_trace_file]			
trace_file_type	constant		IPS

Supplied parameter is:

trace_file_type
The type of trace file for which information is required. Possible values are:

CS File contains tracing on data transferred across the Communications Server for Linux domain between the specified computer and other nodes. This trace type is activated by the **set_cs_trace** command.

TN_SERVER
File contains tracing on the Communications Server for Linux TN server component.

IPS File contains tracing on kernel components for the specified node. This type of trace is activated by the **set_trace_type** or **add_dlc_trace** command.

Returned Parameters

Parameter name	Type	Length
trace_file_type	constant	
dual_files	constant	

```
trace_file_size          decimal
file_name                character      80
file_name_2              character      80
```

If the command executes successfully, Communications Server for Linux returns the following parameters:

trace_file_type
The type of trace file for which information is required (as supplied on the **query_trace_file** command).

dual_files
Specifies whether tracing uses one file or to two files. Possible values are:

YES Tracing uses two files. When the first file reaches the size specified by *trace_file_size*, the second file is cleared, and tracing continues to the second file. When this second file then reaches the size specified by *trace_file_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 *trace_file_size*.

NO Tracing uses one file.

trace_file_size
The maximum size of the trace file. If *dual_files* is set to YES, tracing switches between the two files when the current file reaches this size. If *dual_files* is set to NO, this parameter is ignored; the file size is not limited.

file_name
Name of the trace file, or name of the first trace file if *dual_files* is set to YES.

If no path is included, the file is stored in the default directory for diagnostics files, **/var/opt/ibm/sna**. If a path is included, this path is either a detailed path (starting with a **/** character) or the path relative to the default directory.

file_name_2
Name of the second trace file; this parameter is used only if *dual_files* is set to YES.

If no path is included, the file is stored in the default directory for diagnostics files, **/var/opt/ibm/sna**. If a path is included, this path is either a detailed path (starting with a **/** character) or a path relative to the default directory.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_trace_type

The **query_trace_type** command returns information about the current tracing options for Communications Server for Linux kernel components. For more information about tracing options, refer to *Communications Server for Linux Diagnostics Guide*.

This command does not return information about DLC line tracing. To obtain information about DLC line tracing, use the **query_dlc_trace** command.

This command must be issued to a running node.

Supplied Parameters

```
[query_trace_type]
```

No parameters are supplied for this command.

Returned Parameters

```
Parameter name         Type
trace_flags            constant
truncation_length      decimal
```

If the command executes successfully, Communications Server for Linux returns the following parameters:

trace_flags

The types of tracing currently active. For more information about these trace types, refer to *Communications Server for Linux Diagnostics Guide*.

If tracing is set for all types, one of the following values is returned:

NONE No tracing is active.

ALL Tracing of all types is active.

If tracing is activated for a specific message, one or more of the following values is returned (combined using a + character):

APPC APPC messages are traced.

FM FM messages are traced.

LUA LUA messages are traced.

NOF NOF messages are traced.

MS MS messages are traced.

LLC2 LLC2 messages are traced.

LLI LLI messages are traced.

MAC MAC messages are traced.

SDLC SDLC messages are traced.

NLI NLI messages are traced.

IPDLC Enterprise Extender (HPR/IP) messages are traced.

NDLC Node to DLC messages are traced.

NODE Node internal messages are traced.

SLIM Messages sent between servers in a client/server system are traced.

DGRM Internal control messages between Communications Server for Linux components are traced.

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 length, Communications Server for Linux writes only the start of the message to the trace file and discards the data beyond *truncation_length*. A value of `0` indicates that trace messages are not truncated.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

query_userid_password

The **query_userid_password** command returns information about user ID / password pairs for use with APPC and CPI-C conversation security, or about profiles for a defined user ID and password. This command can be used to obtain information about a specific user ID / password pair or about multiple pairs, depending on the options used.

Supplied Parameters

```
Parameter name          Type         Length     Default
[query_userid_password]
num_entries             decimal                 1
list_options            constant                LIST_INCLUSIVE
user_id                 character    10         (null string)
```

Supplied parameters are:

num_entries
Maximum number of user ID / password pairs for which data should be returned. You can specify `1` to return data for a specific user ID / password pair, a number greater than `1` to return data for multiple user ID / password pairs, or `0` to return data for all user ID / password pairs.

list_options
The position in the list of user ID / password pairs from which Communications Server for Linux begins to return data.

Possible values are:

FIRST_IN_LIST
Start at the first entry in the list

LIST_INCLUSIVE
Start at the entry specified by the *user_id* parameter

LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *user_id* parameter

user_id User ID for which information is required or the user ID to be used as an index into the list of user ID/password pairs. This ID is a type-AE character string. The user ID is ignored if *list_options* is set to FIRST_IN_LIST.

Returned Parameters

Parameter name	Type	Length
user_id	character	10
description	character	31
profile	character	10

(Up to ten profiles can be returned on the *profile* parameter.)

If the command executes successfully, Communications Server for Linux returns the following parameters:

user_id User identifier.

description
A text string describing the user ID and password, as specified in the definition of the user ID and password.

profile Each line is a profile associated with the user.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_USERID
The *list_options* parameter was set to LIST_INCLUSIVE to list all entries starting from the supplied user ID, but the *user_id* parameter value was not valid.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

remove_dlc_trace

The **remove_dlc_trace** command removes DLC line tracing that was previously specified using **add_dlc_trace**. This command can be used to remove all tracing on a resource that is currently being traced, to remove the tracing of certain messages from a resource currently being traced, or to remove all DLC line tracing.

Supplied Parameters

Parameter name [remove_dlc_trace]	Type	Length	Default
resource_type	constant		ALL_DLC_TRACES
resource_name	character	8	(null string)
sidh	hex byte		0
sidl	hex byte		0
odai	constant		NO
message_type	constant		TRACE_ALL

Supplied parameters are:

resource_type
The resource type of the trace entry to remove or modify. Possible values are:

ALL_DLC_TRACES
Remove all DLC tracing options, so that no resources are traced. If this option is specified, the remaining parameters on this command (*resource_name* through *message_type*) are reserved.

ALL_RESOURCES
Remove or modify the tracing options used for tracing all DLCs, ports, link stations, and RTP connections; resources for which DLC_TRACE entries are explicitly defined will continue to be traced.

DLC Remove or modify tracing for the DLC named in *resource_name* and for all ports and link stations that use this DLC.

PORT Remove or modify tracing for the port named in *resource_name* and for all link stations that use this port.

LS Remove or modify tracing for the LS named in *resource_name*.

RTP Remove or modify tracing for the RTP (rapid transport protocol) connection named in *resource_name*.

PORT_DEFINED_LS
Modify tracing for the port named in *resource_name* and its defined link stations.

PORT_IMPLICIT_LS
Modify tracing for the port named in *resource_name* and its implicit link stations.

resource_name
The name of the DLC, port, or link station LS, or RTP connection for which tracing is to be removed or modified.If the name of an RTP connection is specified, this name begins with the `@` character.

If you specify this parameter, *resource_type* must not be set to either `ALL_DLC_TRACES` or `ALL_RESOURCES`.

The following three parameters identify the Local Form Session Identifier for a session on the specified LS. This LFSID is valid only if *resource_type* is set to `LS` and indicates that tracing is to be removed only for messages on this session. The LFSID consists of the following parameters:

sidh The session ID high byte used in identifying the LFSID for a session on an LS.

sidl The session ID low byte used in identifying the LFSID for a session on an LS.

odai The Origin Destination Assignor Indicator used in identifying the LFSID for a session on an LS. Possible values are:

YES The BIND sender is the node containing the secondary link station.

NO The BIND sender is the node containing the primary link station.

message_type
The type of messages for which tracing is being removed for the specified resource or session. To remove tracing for all messages, set this parameter to `TRACE_ALL`. To remove tracing for a specific message, set this parameter to one or more of the following values (combined using a + character):

TRACE_XID
XID messages

TRACE_SC
Session control RUs

TRACE_DFC
Data flow control RUs

TRACE_FMD
Function management data (FMD) messages

TRACE_NLP
Network layer protocol

TRACE_NC
Network connection

TRACE_SEGS
Non-BBIU segments that do not contain an RH

TRACE_CTL
Messages other than MUs and XIDs

For tracing on an RTP connection, the values `TRACE_XID`, `TRACE_NLP`, and `TRACE_CTL` are ignored.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_RESOURCE_TYPE
The value specified in the *resource_type* parameter was not valid.

INVALID_MESSAGE_TYPE
The value specified in the *message_type* parameter was not valid.

INVALID_DLC_NAME
The DLC named in *resource_name* does not have any tracing options set.

INVALID_PORT_NAME
The port named in *resource_name* does not have any tracing options set.

INVALID_LS_NAME
The LS named in *resource_name* does not have any tracing options set.

INVALID_RTP_CONNECTION
The RTP connection named in the *resource_name* parameter does not have any tracing options set.

INVALID_LFSID_SPECIFIED
The LS named in *resource_name* does not have any tracing options set for the specified LFSID.

INVALID_FILTER_TYPE
The *message_type* parameter specified a message type that is not currently being traced for the specified resource.

ALL_RESOURCES_NOT_DEFINED
The *resource_type* parameter was set to ALL_RESOURCES, but a DLC_TRACE entry has not been defined for tracing options on all resources.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

reset_session_limit

The **reset_session_limit** command requests Communications Server for Linux to reset the session limits for a particular LU-LU-mode combination. Sessions may be deactivated as a result of processing this command.

Supplied Parameters

Parameter name	Type	Length	
[reset_session_limit]			
lu_name	character	8	(null string)
lu_alias	character	8	(null string)
plu_alias	character	8	(null string)
fqplu_name	character	17	(null string)
mode_name	character	8	(null string)
mode_name_select	constant		ONE
set_negotiable	constant		NO
responsible	constant		SOURCE
drain_source	constant		NO
drain_target	constant		NO
force	constant		NO

Supplied parameters are:

lu_name
: LU name of the local LU. This name is a type-A character string. To indicate that the LU is identified by its LU alias instead of its LU name, do not specify this parameter.

lu_alias
: LU alias of the local LU. This alias is a character string using any locally displayable characters. It is used only if *lu_name* is not specified.

  If *lu_name* and *lu_alias* are not specified, the command is forwarded to the LU associated with the CP (the default LU).

plu_alias
: LU alias of the partner LU. This alias is a character string using any locally displayable characters. To indicate that the partner LU is defined by its fully qualified LU name instead of its LU alias, do not specify this parameter.

fqplu_name
: Fully qualified name of the partner LU. Specify 3–17 type-A characters that consist of a 1–8 character network name, followed by a period, followed by a 1–8 character partner LU name. For the network name and partner LU name, use only letters, digits `0`–`9`, and special characters `$`, `#`, and `@`.

  This parameter is used only if the *plu_alias* parameter is not specified; it is ignored if *plu_alias* is specified.

mode_name
: Name of the mode for which to reset session limits. This parameter is a type-A character string starting with a letter. It is ignored if *mode_name_select* is set to `ALL`.

mode_name_select
: Selects whether session limits should be reset on a single specified mode, or on all modes between the local and partner LUs. Possible values are:

  ONE Reset session limits on the mode specified by *mode_name*.

  ALL Reset session limits on all modes.

set_negotiable
Specifies whether to reset the maximum negotiable session limit for this LU-LU-mode combination to 0. (The current limit may be the limit specified for the mode, or may have been changed by **initialize_session_limit** or **change_session_limit**.) Possible values are:

YES Reset the maximum negotiable session limit for this LU-LU-mode combination to 0 (so that sessions cannot be activated until the limit is changed by **initialize_session_limit**).

NO Leave the maximum negotiable session limit unchanged.

responsible
Indicates whether the source (local) or target (partner) LU is responsible for deactivating sessions after the session limit is reset. Possible values are:

SOURCE The local LU is responsible for deactivating sessions.

TARGET The partner LU is responsible for deactivating sessions.

drain_source
Specifies whether the source LU satisfies waiting session requests before deactivating a session. Possible values are:

YES Waiting session requests are satisfied.

NO Waiting session requests are not satisfied.

drain_target
Specifies whether the target LU satisfies waiting session requests before deactivating a session. Possible values are:

YES Waiting session requests are satisfied.

NO Waiting session requests are not satisfied.

force Specifies whether to set session limits to 0 even if CNOS negotiation fails. Possible values are:

YES Set session limits to 0.

NO Do not set session limits to 0 even if CNOS negotiation fails.

Returned Parameters

If the command executes successfully, Communications Server for Linux returns the following parameters:

primary_rc
OK

secondary_rc
Possible values are:

AS_SPECIFIED
The command executed successfully. The session limits were changed as specified.

FORCED The session limits were set to 0 even though CNOS negotiation failed.

AS_NEGOTIATED
The session limits were changed, but one or more values were negotiated by the partner LU.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

EXCEEDS_MAX_ALLOWED
A Communications Server for Linux internal error occurred.

INVALID_LU_ALIAS
The *lu_alias* parameter value did not match any defined local LU alias.

INVALID_LU_NAME
The *lu_name* parameter value did not match any defined local LU name.

INVALID_MODE_NAME
The *mode_name* parameter value did not match any defined mode name.

INVALID_PLU_NAME
The *fqplu_name* parameter value did not match any defined partner LU name.

INVALID_MODE_NAME_SELECT
The *mode_name_select* parameter was not set to a valid value.

INVALID_DRAIN_SOURCE
The *drain_source* parameter was not set to a valid value.

INVALID_DRAIN_TARGET
The *drain_target* parameter was not set to a valid value.

INVALID_FORCE
The *force* parameter was not set to a valid value.

INVALID_RESPONSIBLE
The *responsible* parameter was not set to a valid value.

INVALID_SET_NEGOTIABLE
The *set_negotiable* parameter was not set to a valid value.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc

MODE_RESET
No sessions are currently active for this LU-LU-mode combination. Use **initialize_session_limit** instead of **reset_session_limit** to specify the limits.

Other Conditions

If the command does not execute because other conditions exist, Communications Server for Linux returns the following parameters:

primary_rc

ALLOCATION_ERROR
A session could not be allocated because of a condition that requires action. Check the log files for messages indicating the reason for the failure, and take any action required. Do not attempt to retry the command until the condition has been corrected.

secondary_rc

ALLOCATION_FAILURE_NO_RETRY
A session could not be allocated because of a condition that requires action. Check the *sense_data* parameter and logged messages to determine the reason for the failure, and take any action required. Do not attempt to retry the command until the condition has been corrected.

sense_data
The SNA sense data associated with the allocation failure.

primary_rc

CONV_FAILURE_NO_RETRY
The session limits could not be changed because of a condition that requires action (such as a configuration mismatch or a session protocol error). Check the Communications Server for Linux log file for information about the error condition and correct it before retrying this command.

primary_rc

CNOS_PARTNER_LU_REJECT
The command failed because Communications Server for Linux failed to negotiate the session limits with the partner. Check configuration at both the local LU and partner LU.

secondary_rc

CNOS_COMMAND_RACE_REJECT
The command failed because the specified mode was being accessed by another administration program (or internally by the Communications Server for Linux software) for session activation or deactivation, or for session limit processing. Retry the command.

Appendix A, “Common Return Codes from snaadmin Commands,” on page 589 lists combinations of primary and secondary return codes that are common to all commands.

set_buffer_availability

The **set_buffer_availability** command specifies the amount of STREAMS buffer space that Communications Server for Linux can use at any one time. This information enables the node to make efficient use of the buffer space available, and enables you to ensure that buffer space is available for other processes on the Linux computer.

Supplied Parameters

```
Parameter name     Type
[set_buffer_availability]
buf_avail          decimal
```

Supplied parameter is:

buf_avail
The maximum amount of STREAMS buffer space available, in bytes.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

set_central_logging

The **set_central_logging** command specifies whether Communications Server for Linux log messages are sent from all servers to a central file, or to a separate file on each server. For more information about log files, see "set_log_file" on page 553.

This command must be issued without specifying a node name.

Supplied Parameters

```
Parameter name          Type        Length    Default
[set_central_logging]
enabled                 constant              YES
```

Supplied parameter is:

enabled Specifies whether to enable or disable central logging. Possible values are:

YES Enable central logging. All log messages are sent to a single central file on the node that is currently the central logger.

NO Disable central logging. Log messages from each server are sent to a file on that server (specified using **set_log_file**).

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc

PARAMETER_CHECK

secondary_rc

NOT_CENTRAL_LOGGER
The command was issued to a specific node. It must be issued without specifying a node name.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

set_cs_trace

The **set_cs_trace** command specifies tracing options for data sent between computers in the Communications Server for Linux domain. For more information about tracing options, refer to *Communications Server for Linux Diagnostics Guide*.

This command can be issued from an AIX or Linux client. The command must run with the userid `root`, or with a userid that is a member of the `sys` group (AIX) or `sna` group (Linux).

This command must be issued to a running node, unless it is issued from a client.

On Windows clients, client/server tracing is controlled by options in the Windows Registry. For more information, refer to *Communications Server for Linux Diagnostics Guide*.

Supplied Parameters

Parameter name	Type	Length	Default
[set_cs_trace]			
dest_sys	character	128	(null string)
trace_flags	constant		NONE
trace_direction	constant		CS_BOTH

Supplied parameters are:

dest_sys
The server name for which tracing is required. This name is a string of locally displayable characters.

To manage tracing on messages flowing between the computer to which this command is issued (either the local computer or one identified by the **-n** option on the **snaadmin** program) and one other node in the domain, specify the name of the other node; tracing on messages flowing to and from other computers in the domain will be unchanged. You can issue two **set_cs_trace** commands to activate tracing between the same target computer and two different destination servers.

If the server name includes a **.** (period) character, Communications Server for Linux assumes that it is a fully-qualified name; otherwise it performs a DNS lookup to determine the server name.

To manage tracing on messages flowing between the computer to which this command is issued (either the local computer or one identified by the **-n** option on the **snaadmin** program) and all other nodes in the domain, do not specify this parameter. The options you specify on this command override any previous settings for tracing to specific computers (identified by *dest_sys* on the previous **set_cs_trace** commands).

trace_flags
The types of tracing required. For more information about these trace types, refer to *Communications Server for Linux Diagnostics Guide*.

To set tracing for all types use one of the following values:

NONE Do not activate tracing for any type of message.

ALL Activate tracing for all types of messages.

To activate tracing on specific message types, select one or more of the following values (combined using a + character):

CS_ADMIN_MSG
Trace internal messages relating to client/server topology

CS_DATAGRAM
Trace datagram messages

CS_DATA
Trace data messages

trace_direction
Specifies the direction or directions in which tracing is required. This parameter is ignored if *trace_flags* is set to NONE. Possible values are:

CS_SEND
Trace messages flowing from the target computer to the computer defined by *dest_sys*.

CS_RECEIVE
Trace messages flowing from the computer defined by *dest_sys* to the target computer.

CS_BOTH
Trace messages flowing in both directions.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

NAME_NOT_FOUND
The server specified by the *dest_sys* parameter was not valid or was not started.

LOCAL_SYSTEM
The server specified by the *dest_sys* parameter is the same as the target node to which this command was issued.

INVALID_TRC_DIRECTION
The *trace_direction* parameter was not set to a valid value.

INVALID_TARGET
The command was issued on a standalone server. This command can only be issued on a client/server system.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

set_global_log_type

The **set_global_log_type** command specifies the types of information that Communications Server for Linux records in log files. It specifies default values that are used on all computers; you can then use **set_log_type** (or, for a Windows client, options in the Windows Registry)to override these defaults on a particular computer.For more information about log files, see "set_log_file" on page 553.

Communications Server for Linux always logs messages for problem events; you can specify whether to log messages for exception and audit events. For more information logging messages, refer to the *Communications Server for Linux Diagnostics Guide*.

This command must be issued without specifying a node name.

Supplied Parameters

Parameter name [set_global_log_type]	Type	Length	Default
audit	constant		LEAVE_UNCHANGED
exception	constant		LEAVE_UNCHANGED
succinct_audits	constant		LEAVE_UNCHANGED
succinct_errors	constant		LEAVE_UNCHANGED

Supplied parameters are:

audit Specify whether to record audit messages. Possible values are:

YES Record audit messages.

NO Do not record audit messages.

LEAVE_UNCHANGED
Leave audit logging unchanged from the existing definition. (Communications Server for Linux initially sets *audit* to NO.)

exception
Specifies whether to record exception messages. Possible values are:

YES Record exception messages.

NO Do not record exception messages.

LEAVE_UNCHANGED
Leave exception logging unchanged from the existing definition. (Communications Server for Linux initially sets *exception* to YES.)

succinct_audits
Specifies whether to use succinct logging or detailed logging in the audit log file. Possible values are:

YES Use succinct logging in the audit log file. Each message in the log file contains a summary of the message header information (such as the message number, log type, and system name), and the message text string and parameters. To obtain more details about the cause of the log and any action required, use the **snahelp** utility.

NO Use detailed logging in the audit log file. Each message in the log file includes a full listing of the message header information, the message text string and parameters, and additional information about the cause of the log and any action required.

LEAVE_UNCHANGED
Use the option (succinct logging or detailed logging) specified on the previous **set_global_log_type** command. (Before a **set_global_log_type** command has been issued, Communications Server for Linux initially sets *succinct_audits* to YES.)

If you are using central logging, the choice of succinct or detailed logging for messages from all computers is determined by setting 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_errors
Specifies whether to use succinct logging or detailed logging in the error log file; this applies to both exception logs and problem logs. Possible values are:

YES Use succinct logging in the error log file. Each message in the log file contains a summary of the message header information (such as the message number, log type, and system name), and the message text string and parameters. To obtain more details about the cause of the log and any action required, use the **snahelp** utility.

NO Use detailed logging in the error log file. Each message in the log file includes a full listing of the message header information, the message text string and parameters, and additional information about the cause of the log and any action required.

LEAVE_UNCHANGED
Use the option (succinct logging or detailed logging) specified on the previous **set_global_log_type** command. (Before a **set_global_log_type** command has been issued, Communications Server for Linux initially sets *succinct_audits* to YES.)

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

NOT_CENTRAL_LOGGER
The command was issued to a specific node. It must be issued without specifying a node name.

INVALID_SUCCINCT_SETTING
The *succinct_audits* or *succinct_errors* parameter was not set to a valid value.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

set_kernel_memory_limit

The **set_kernel_memory_limit** command specifies a limit on the amount of kernel memory that Communications Server for Linux can use at any one time. This limit enables you to ensure that memory is available for other processes on the Linux computer.

You can also specify the kernel memory limit when starting the Communications Server for Linux software; for more information, refer to the *Communications Server for Linux Administration Guide*. If any limit was specified when starting the Communications Server for Linux software, this command overrides that limit.

Supplied Parameters

```
Parameter name          Type
[set_kernel_memory_limit]
limit                   decimal
```

Supplied parameter is:

limit The maximum amount of kernel memory that Communications Server for Linux uses at any one time, in bytes. If a Communications Server for Linux component attempts to allocate kernel memory that would take the total amount of memory currently allocated above this limit, the allocation attempt will fail.

To remove the limit set by a previous **set_kernel_memory_limit** command, specify the value `0`.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

set_log_file

The **set_log_file** command manages a file that Communications Server for Linux uses to record log messages. It enables you to do the following:

- Specify a file used to record log messages (audit, error, or usage logs), and the backup file (to which log information is copied).
- Specify the maximum log file size (when the log file reaches this size, Communications Server for Linux copies log information to the backup file and resets the log file).
- Copy the current contents of the log file to the backup file, and optionally delete the current file.

You can record audit log and error log messages in separate files, or record both types of messages in the same file.

If you are using central logging, as defined by the **set_central_logging** command, this command must be issued to the node that is acting as the central logger. Otherwise you can issue it to each node separately in order to specify a different log file on each node.

This command can be issued from an AIX or Linux client. The command must run with the userid `root`, or with a userid that is a member of the `sys` group (AIX) or `sna` group (Linux).

Supplied Parameters

Parameter name	Type	Length	Default
[set_log_file]			
log_file_type	constant		ERROR
action	constant		NO_FILE_ACTION
file_name	character	80	(null string)
backup_file_name	character	80	(null string)
file_size	decimal		0

Supplied parameters are:

log_file_type
The type of log file to be used. Possible values are:

AUDIT Audit log file (record audit messages only)

ERROR Error log file (record problem and exception messages)

USAGE Usage log file (record information on current and peak usage of Communications Server for Linux resources).

To record both audit and error messages in the same file, issue two **set_log_file** commands for the same file name, specifying `AUDIT` for the *log_file_type* of one command and `ERROR` for the *log_file_type* of the other.

action The action to be taken on the log file. Specify one of the following values:

NO_FILE_ACTION
Use the file specified in the *file_name* parameter as the log file, and the file specified in the *backup_file_name* parameter as the backup file. After this command completes successfully, all log messages of the type defined by *log_file_type* are written to the new log file. If a log file was used before this command is issued, the log file is left unchanged.

DELETE_FILE
Delete the contents of the current log file.

BACKUP_FILE
Copy the contents of the current log file to the backup file, and then delete the contents of the current file.

file_name
Name of the new log file.

To create the file in the default directory for diagnostics files, **/var/opt/ibm/sna**, specify the file name with no path. To create the file in a different directory, specify either a full path or the path relative to the default directory. If you include the path, ensure that it is a valid path (either a path relative to the application's working directory or a full path) on any computer to which this command is issued.

This parameter is an ASCII string of 1–80 characters. To continue logging to the file specified on a previous **set_log_file** command, do not specify this parameter. The initial defaults, before any **set_log_file** command has been issued, are **/var/opt/ibm/sna/sna.err** for the error log file, **/var/opt/ibm/sna/sna.aud** for the audit log file, and **/var/opt/ibm/sna/sna.usage** for the usage log file.

backup_file_name
Name of the backup log file. When the log file reaches the size specified by the *file_size* parameter, Communications Server for Linux copies the current contents to the backup file and then clears the log file. You can also request a backup at any time using the *action* parameter.

To create the file in the default directory for diagnostics files, **/var/opt/ibm/sna**, specify the file name with no path. To create the file in a different directory, specify either a full path or the path relative to the default directory. If you include the path, ensure that it is a valid path (either a path relative to the application's working directory or a full path) on any computer to which this command is issued.

This parameter is an ASCII string of 1–80 characters, terminated with a null character (binary zero). To continue using the backup file specified on a previous **set_log_file** command, do not specify this parameter. The initial defaults, before any **set_log_file** command has been issued, are **/var/opt/ibm/sna/bak.err** for the error log file, **/var/opt/ibm/sna/bak.aud** for the audit log file, and **/var/opt/ibm/sna/bak.usage** for the usage log file.

file_size
The maximum size of the log file specified by *log_file_type*. When a message written to the file causes the file size to exceed this limit, Communications Server for Linux copies the current contents of the log file to the backup log file and clears the log file. The maximum amount of disk space taken up by log files is approximately twice *file_size*.

To continue using the file size specified on a previous **set_log_file** command, do not specify this parameter. The initial default value, before any **set_log_file** command has been issued, is 1,000,000 bytes. A value of `0` indicates "continue using the existing file size" and not "no limit."

You may need to increase the size of the audit and error log filesaccording to the size of the Communications Server for Linux client/server network, to allow for the volume of log information generated in larger systems. In particular, consider increasing the log file size to allow for the following:

- Accommodating large numbers of clients or users (because a single communications link failure may result in large numbers of logs on the server relating to session failures)
- Activating audit logging as well as exception logging
- Using central logging instead of distributed logging
- Using detailed logging instead of succinct logging

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc

PARAMETER_CHECK

secondary_rc

INVALID_FILE_ACTION
The *action* parameter was not set to a valid value.

INVALID_FILE_TYPE
The *log_file_type* parameter was not set to a valid value.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

set_log_type

The **set_log_type** command specifies the types of information that Communications Server for Linux records in log files on a particular server. This command can be used to override the default settings specified on **set_global_log_type**, or to remove the override so that this server reverts to using the default settings. For more information about log files, see "set_log_file" on page 553.

Communications Server for Linux always logs messages for problem events; you can specify whether to log messages for exception and audit events. For more information about logging messages, refer to the *Communications Server for Linux Diagnostics Guide*.

This command can be issued from an AIX or Linux client. The command must run with the userid `root`, or with a userid that is a member of the `sys` group (AIX) or `sna` group (Linux).

Supplied Parameters

Parameter name [set_log_type]	Type	Length	Default
override	constant		YES
audit	constant		LEAVE_UNCHANGED
exception	constant		LEAVE_UNCHANGED
succinct_audits	constant		LEAVE_UNCHANGED
succinct_errors	constant		LEAVE_UNCHANGED

Supplied parameters are:

override
Specifies whether to override the global log types specified on **set_global_log_type** or to revert to using global log types. Possible values are:

YES Override the global log types. The log types to be used on this server are specified by the *audit* and *exception* parameters, and the choice of succinct or detailed logging is specified by the *succinct_** parameters.

NO Revert to using the global log types. The *audit*, *exception*, and *succinct_** parameters are ignored.

audit Specify whether to record audit messages. (Communications Server for Linux initially sets *audit* to `NO`.) Possible values are:

YES Record audit messages.

NO Do not record audit messages.

LEAVE_UNCHANGED
Leave audit logging unchanged from the existing definition.

exception
Specify whether to record exception messages. (Communications Server for Linux initially sets *exception* to `YES`.) Possible values are:

YES Record exception messages.

NO Do not record exception messages.

LEAVE_UNCHANGED
Leave exception logging unchanged from the existing definition.

succinct_audits
Specifies whether to use succinct logging or detailed logging in the audit log file on this server. Possible values are:

YES Use succinct logging in the audit log file. Each message in the log file contains a summary of the message header information (such as the message number, log type, and system name), and the message text string and parameters. To obtain more details about the cause of the log and any action required, use the **snahelp** utility.

NO Use detailed logging in the audit log file. Each message in the log file includes a full listing of the message header information, the message text string and parameters, and additional information about the cause of the log and any action required.

LEAVE_UNCHANGED
Leave succinct logging or detailed logging unchanged from the existing definition.

If you are using central logging, the choice of succinct or detailed 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_errors
Specifies whether to use succinct logging or detailed logging in the error log file on this server; this applies to both exception logs and problem logs. Possible values are:

YES Use succinct logging in the error log file. Each message in the log file contains a summary of the message header information (such as the message number, log type, and system name), and the message text string and parameters. To obtain more details about the cause of the log and any action required, use the **snahelp** utility.

NO Use detailed logging in the error log file. Each message in the log file includes a full listing of the message header information, the message text string and parameters, and additional information about the cause of the log and any action required.

LEAVE_UNCHANGED
Leave succinct logging or detailed logging unchanged from the existing definition.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

INVALID_SUCCINCT_SETTING
The *succinct_audits* or *succinct_errors* parameter was not set to a valid value.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

set_tn_server_trace

The **set_tn_server_trace** command specifies tracing options for the Communications Server for Linux TN server feature.

This command must be issued to a running node.

Supplied Parameters

```
Parameter name        Type          Length    Default
[set_tn_server_trace]
trace_flags           constant                NONE
```

Supplied parameters are:

trace_flags

The types of tracing required. To set tracing for all types of messages, specify one of the following values:

NONE Do not activate tracing for any type of message.

ALL Activate tracing for all types of messages.

To activate tracing on specific message types, select one or more of the following values (combined using a + character):

TCP Trace messages between TN server and TN3270 clients (TCP/IP interface tracing).

FMAPI Trace internal control messages and messages between TN server and TN3270 clients, in internal format (node interface tracing).

CFG Trace messages relating to the configuration of TN server (configuration message tracing).

NOF Trace internal node operator function (NOF) requests made by TN server.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

set_trace_file

The **set_trace_file** command specifies the name of the file that Communications Server for Linux uses to record trace data.

If you issue a second **set_trace_file** command specifying a new file name for the same file type, all subsequent trace data will be written to the new file; the existing file is not removed, but further information will not be written to it.

To reset the current trace file or files while tracing is active (the existing contents of the file are discarded, but any subsequent tracing is written to the same file), issue a **set_trace_file** command specifying the same trace file name and backup file name as the files that are currently being used.

This command may be issued to a running node, or (for client/server trace files only) to a Remote API Client on AIX or Linuxg. To issue the command to a client computer, use the **snaadmin** program on the client computer without specifying a node name.

On Windows clients, tracing is controlled by options in the Windows Registry. For more information, refer to *Communications Server for Linux Diagnostics Guide*.

Supplied Parameters

Parameter name	Type	Length	Default
[set_trace_file]			
trace_file_type	constant		IPS
dual_files	constant		LEAVE_UNCHANGED
trace_file_size	decimal		1000000
file_name	character	80	(null string)
file_name_2	character	80	(null string)

Supplied parameters are:

trace_file_type
The type of trace file. Possible values are:

CS File contains tracing on data transferred across the Communications Server for Linux domain between the specified computer and other nodes. This type of tracing is activated by the **set_cs_trace** command.

TN_SERVER
File contains tracing on the Communications Server for Linux TN server component.

IPS File contains tracing on kernel components for the specified node. This type of tracing is activated by the **set_trace_type** or **add_dlc_trace** command.

dual_files
Specifies whether tracing uses one file or to two files. Possible values are:

YES Tracing uses two files. When the first file reaches the size specified by *trace_file_size*, the second file is cleared, and tracing continues to

the second file. When this second file reaches the size specified by *trace_file_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 *trace_file_size*.

NO Tracing uses one file.

LEAVE_UNCHANGED
Leave the *dual_files* setting unchanged from the existing definition. (When the Communications Server for Linux software is started, the initial default is to use two files.)

trace_file_size
The maximum size of the trace file, in bytes. To continue using the existing trace file size definition, specify `0`.

If *dual_files* is set to `YES`, tracing switches between the two files when the current file reaches this size. If *dual_files* is set to `NO`, this parameter is ignored; the file size is not limited.

You may need to increase the size of the trace files according to the size of the Communications Server for Linux client/server network to allow for the volume of trace information generated in larger systems. Consider increasing the trace file size on a server to allow for large numbers of clients or users accessing the server.

file_name
Name of the trace file, or the name of the first trace file if *dual_files* is set to `YES`. To continue using the file name specified on a previous **set_trace_file** command, do not specify this parameter.

To create the file in the default directory for diagnostics files, **/var/opt/ibm/sna**, specify the file name with no path. To create the file in a different directory, specify either a full path or the path relative to the default directory. If you include the path, ensure that it is a valid path (either a path relative to the application's working directory or a full path) on any computer to which this command is issued.

file_name_2
Name of the second trace file; this parameter is used only if *dual_files* is set to `YES`. To continue using the file name specified on a previous **set_trace_file** command, do not specify this parameter.

To create the file in the default directory for diagnostics files, **/var/opt/ibm/sna**, specify the file name with no path. To create the file in a different directory, specify either a full path or the path relative to the default directory. If you include the path, ensure that it is a valid path (either a path relative to the application's working directory or a full path) on any computer to which this command is issued.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_FILE_NAME
The *file_name* or *file_name_2* parameter was not set to a valid Linux file name, or *file_name_2* was not specified when changing from a single trace file to dual trace files.

INVALID_FILE_TYPE
The *trace_file_type* parameter was not set to a valid value.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

set_trace_type

The **set_trace_type** command specifies tracing options for Communications Server for Linux kernel components. You can use this command to specify the state of tracing (on or off) at all interfaces or to turn tracing on or off at specific interfaces (leaving tracing at other interfaces unchanged). For more information about tracing options, refer to the *Communications Server for Linux Diagnostics Guide*.

To control DLC line tracing, use the **add_dlc_trace** command. The truncation length specified on this command also applies to DLC tracing, but the tracing options on this command do not apply to DLC tracing.

This command must be issued to a running node.

Supplied Parameters

Parameter name [set_trace_type]	Type	Length	Default
trace_flags	constant		NONE
truncation_length	decimal		1024
init_flags	constant		YES
set_flags	constant		YES

Supplied parameters are:

trace_flags
The types of tracing required. For more information about these trace types, refer to *Communications Server for Linux Diagnostics Guide*.

If *init_flags* is set to YES, select the values corresponding to the interfaces where you want tracing to be active, and do not select the values corresponding to the interfaces where you want it to be inactive. If

init_flags is set to NO, select the values corresponding to the interfaces where you want to change the state of tracing.

To set tracing for all types of messages use one of the following values:

NONE Do not activate tracing for any type of messages.

ALL Activate tracing for all types of messages.

To set tracing for a specific interface, use one or more of the following values (combined using + characters).

APPC Trace APPC messages

LUA Trace LUA messages

NOF Trace NOF messages

MS Trace MS messages

LLC2 Trace LLC2 messages

LLI Trace LLI messages

MAC Trace MAC messages

SDLC Trace SDLC messages (note that this option also provides additional detail in SDLC line tracing)

NLI Trace NLI messages

IPDLC Trace Enterprise Extender (HPR/IP) messages

NDLC Trace node to DLC messages

NODE Trace node internal messages

SLIM Trace messages sent between servers in a client/server system

DGRM Trace internal control messages between Communications Server for Linux components

truncation_length

The maximum length, in bytes, of the information written to the trace file for each message. This value must be at least 256.

If a message is longer than this value, Communications Server for Linux writes only the start of the message to the trace file and discards the data beyond *truncation_length*. Truncation enables you to record the most important information for each message but avoid filling up the file with long messages.

To specify no truncation (all the data from each message is written to the file), set this parameter to 0.

init_flags

Specifies whether to initialize tracing (define the tracing state at all interfaces) or to change the state of tracing at one or more interfaces (leaving the others unchanged). Possible values are:

YES Initialize tracing. The *trace_flags* parameter defines the required state of tracing at all interfaces.

NO Change the state of tracing. The *trace_flags* parameter defines the interfaces where tracing is to be activated or deactivated; other interfaces will not be affected.

set_flags
If *init_flags* is set to NO, this parameter specifies whether to activate or deactivate tracing at the requested interfaces. Possible values are:

YES Activate tracing at the interfaces specified by the *trace_flags* parameter.

NO Deactivate tracing at the interfaces specified by the *trace_flags* parameter.

If *init_flags* is set to YES, this parameter is ignored.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

INVALID_TRUNC_LEN
The *truncation_length* parameter specified a length of less than 256 bytes.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

start_dlc

The **start_dlc** command activates a DLC.

When this command is issued, the associated node is activated automatically if it is not already active.

If this command does not return an error message, this indicates only that the command was issued successfully. The command does not wait for the DLC to initialize and therefore does not return error return codes if the initialization of the DLC fails. DLC initialization failures are reported by messages written to the error log file.

Supplied Parameters

```
Parameter name          Type            Length
[start_dlc]
dlc_name                character       8
```

Supplied parameters are:

dlc_name
Name of the DLC to be started. This must match the name of a defined DLC.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

INVALID_DLC
The name supplied for the *dlc_name* parameter was not the name of a defined DLC.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc

DLC_DEACTIVATING
The specified DLC has already been started, and is being deactivated.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

start_internal_pu

The **start_internal_pu** command requests DLUR to initiate SSCP-PU session activation for a previously defined local PU that is served by DLUR.

This command must be issued to a running node.

Supplied Parameters

Parameter name	Type	Length	Default
[start_internal_pu]			
pu_name	character	8	
dlus_name	character	17	(null string)
bkup_dlus_name	character	17	(null string)

Supplied parameters are:

pu_name
: Name of the internal PU to be started. This PU must have been previously defined using **define_internal_pu**. The name is a type-A character string starting with a letter.

dlus_name
: Name of the DLUS node that DLUR will contact to solicit SSCP-PU session activation for the given PU. Specify 3–17 type-A characters, consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character DLUS node name.

  To use the DLUS specified in **define_internal_pu**, or the global default specified in **define_dlur_defaults** if no DLUS was specified in **define_internal_pu**, do not specify this parameter.

bkup_dlus_name
: Name of the DLUS node that DLUR will store as the backup DLUS for the given PU. Specify 3–17 type-A characters, consisting of a 1–8 character network name, followed by a period, followed by a 1–8 character backup DLUS name.

  To use the backup DLUS specified in **define_internal_pu**, or the global backup default specified in **define_dlur_defaults** if no backup DLUS was specified in **define_internal_pu**, do not specify this parameter.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
: PARAMETER_CHECK

secondary_rc
: Possible values are:

  INVALID_DLUS_NAME
  : The *dlus_name* parameter contained a character that was not valid or was not in the correct format.

  INVALID_BKUP_DLUS_NAME
  : The *bkup_dlus_name* parameter contained a character that was not valid or was not in the correct format.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
: STATE_CHECK

secondary_rc
: Possible values are:

NO_DEFAULT_DLUS_DEFINED
: A DLUS name was not specified either on this command or on **define_internal_pu**, and there is no default DLUS defined (because **define_dlur_defaults** has not been issued).

PU_NOT_DEFINED
: The supplied PU name was not the name of an internal PU defined using **define_internal_pu**.

PU_ALREADY_ACTIVATING
: The PU is already being activated.

PU_ALREADY_ACTIVE
: The PU has already been activated.

Function Not Supported

If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameters:

primary_rc

FUNCTION_NOT_SUPPORTED
: The node does not support DLUR; support is defined by the *dlur_support* parameter on the **define_node** command.

secondary_rc
: (This parameter is not used.)

Other Conditions

If the command does not execute because other conditions exist, Communications Server for Linux returns the following parameters:

primary_rc
: UNSUCCESSFUL

secondary_rc
: Possible values are:

DLUS_REJECTED
: The DLUS rejected the session activation request.

DLUS_CAPS_MISMATCH
: The configured DLUS name was not a DLUS node.

PU_FAILED_ACTPU
: The local node rejected a message from the DLUS. This can be caused by an internal error, a resource shortage, or a problem with the received message; check the Communications Server for Linux log files for messages providing more information.

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

start_ls

The **start_ls** command is usually used to start an inactive link station (LS). Alternatively, the command can be used to leave the LS inactive but specify that it can be automatically activated by Communications Server for Linux when required or activated by the remote system.

If this command is used to activate the LS, the associated port, DLC, and node are activated automatically if they are not already active.

Note: If the LS is a leased SDLC link or a QLLC PVC link, it must be activated by the remote system as well as by Communications Server for Linux. You are recommended to define the LS to be activated when the node is started and to be reactivated automatically after failures, to ensure that the link is always available; see "define_sdlc_ls" on page 162 or "define_qllc_ls" on page 135 for more information.

Supplied Parameters

```
Parameter name     Type          Length    Default
[start_ls]
ls_name            character     8
enable             constant                ACTIVATE
```

Supplied parameters are:

ls_name
Name of the link station to be started. This LS must already have been previously defined.

enable Specifies the action to be taken for the LS.

To start the LS, set this parameter to `ACTIVATE`.

To leave the LS inactive but specify that it can be activated (either by Communications Server for Linux or by the remote system) when required, specify one or both of the following values (combined using a logical OR):

`AUTO_ACT`
The LS can be automatically activated by Communications Server for Linux when required for a session. This value should be used only when the LS is defined to automatically activated (*auto_act_supp* in the LS definition is set to `YES`). This action re-enables auto-activation after the LS has been manually stopped using **stop_ls**.

`REMOTE_ACT`
The LS can be activated by the remote system. This value does not alter the defined value of the *disable_remote_act* parameter in the LS definition; when the LS is next stopped, it will return to the defined setting.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_LINK_NAME_SPECIFIED
The *ls_name* parameter was not the name of a defined LS.

INVALID_LINK_ENABLE
The *enable* parameter was not set to a valid value.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

ACTIVATION_LIMITS_REACHED
The LS cannot be started because the outbound link activation limit has been reached.

PARALLEL_TGS_NOT_SUPPORTED
A link to the remote system is already active. The adjacent node does not support parallel transmission groups.

LINK_DEACT_IN_PROGRESS
The specified LS is currently being deactivated. You cannot start it until deactivation is complete.

ALREADY_STARTING
The specified LS is already starting.

Unsuccessful

If the command does not execute successfully because the SNA subsystem on the remote computer cannot be contacted, Communications Server for Linux returns the following parameters:

primary_rc
LS_FAILURE

secondary_rc
Possible values are:

PARTNER_NOT_FOUND
No response was received from the port associated with this LS. For Token Ring, Ethernet: check that the *mac_address* parameter in the LS definition is correct.

ERROR The connection to the remote computer could not be established.

This may be because the SNA subsystem on the remote computer is not started. For link types other than LAN types (Token Ring, Ethernet), it may also indicate that Communications Server for Linux could not find a remote computer matching the supplied addressing information.

Other Conditions

If the command does not execute because other conditions exist, Communications Server for Linux returns the following parameters:

primary_rc
CANCELLED

secondary_rc
Possible values are:

NO_SECONDARY_RC
A **stop_ls** command was issued before the **start_ls** command had completed. The **start_ls** command was canceled.

LINK_DEACTIVATED
The DLC or port used by the LS was stopped before the **start_ls** command had completed. The **start_ls** command was canceled.

start_port

The **start_port** command requests the activation of a port.

When this command is issued, the associated DLC and node are activated automatically if they are not already active.

Supplied Parameters

Parameter name	Type	Length
[start_port]		
port_name	character	8

Supplied parameters are:

port_name
Name of the port to be started. The port must already have been defined.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

INVALID_PORT
The *port_name* parameter was not the name of a defined port.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

DUPLICATE_PORT
The specified port has already been started.

STOP_PORT_PENDING
The specified port is currently being deactivated. You cannot start the port until deactivation is complete.

Other Conditions

If the command does not execute because other conditions exist, Communications Server for Linux returns the following parameters:

primary_rc
CANCELLED

secondary_rc

NO_SECONDARY_RC
A **stop_port** command was issued before this command had completed; the **start_port** command was canceled.

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

status_all

The **status_all** command returns status information about all resources. This command returns all status information that is returned by other **status_*** commands. For more details about status information returned by each **status_*** command, see "status_connectivity" on page 573, "status_dependent_lu" on page 574, "status_dlur" on page 577, "status_lu62" on page 578, and "status_node" on page 579.

Supplied Parameters

```
Parameter name              Length
[status_all]
```

No parameters are supplied for this command.

Returned Information

Communications Server for Linux returns status information for all status categories that are available with the other **status_*** commands. If the node does not support dependent LU requester (DLUR), no DLUR status is returned.

The following example illustrates the information returned for the **status_all** command.

The amount of information returned depends on the width of your display; Communications Server for Linux attempts to determine this using the COLUMNS environment variable, and uses a default value of 80 if COLUMNS is not set. The *Description* text may be omitted or truncated if there is not enough room to display it.

status_all

```
Returned Information for the status_all Command
Communications Server for Linux         Thursday 9:30:20        16 May 1996
================================================================================
Node           Status         Role           Description
--------------------------------------------------------------------------------
george         Active         Master         Network node
--------------------------------------------------------------------------------
DLC       Port      LS        PU        Type Status         Description
--------------------------------------------------------------------------------
TOKEN0                                 TR   Active         /dev/tr0
          TRP1                         TR   Inactive       First port
                    TRL0      PU0       TR   Inactive       Link to host
          TRP2                         TR   Active         Second port
                    TRL1      AS400     TR   Active         Link to AS/4
                    TRL2      PUNAME    TR   Stopping       Link to othe
ETH0                                   Eth  Active         Another DLC
          ETHER0                       Eth  Active         My Ethernet
                    ETH0      PU5       Eth  On demand      Link for app
                    DOWN      PU6       Eth  Inactive       Downstream t
--------------------------------------------------------------------------------
PU        LS        NAU  LU        LU type   Status         Description
--------------------------------------------------------------------------------
PU0       TRL0                              Inactive       Link to host
                     3  LU1       DISPLAY   Inactive       Freds Display
                     4  LU2       PRINTER   Inactive       Fred's printe
                    17  DEPLU1    LU62      Inactive       Used by APP1
PU5       ETH0                              SSCP           Link for appl
                    11  LU4       OTHER     Inactive       Used for TN32
                    12  LU5       DISPLAY   Active         Model 5 displ
                                            3270 display user: liz
                                            Computer: george
                    13  DEPLU2    LU62      Active         Used by APP2
                                            Partner LU: APPN.PARTNER
                                            Mode: MODE1
PU6       DOWN                              Inactive       Downstream to
                    99  DSLU99    PRINTER   Inactive       DS for the ot
--------------------------------------------------------------------------------
DLUR PU  LU         DLUS           PLU                Description
--------------------------------------------------------------------------------
DSPU1 (Downstream)  APPN.DLUS
         DLU1       Inactive        Inactive
         DLU2       APPN.DLUS       APPN.PLU2
PU0                 APPN.DLUS                          Host in Naples
         DLU0       APPN.DLUS       APPN.PLU0          Display mod2
PU2                                 Inactive           Host in Athens
         DLU3       Inactive        Inactive           Display mod2
--------------------------------------------------------------------------------
LU        LU Alias     Machine       Partner LU             Mode         Session count
--------------------------------------------------------------------------------
GEORGE    GEORGE                                            Inactive
FRED      FALIAS       client1        APPN.AS400            CPSVGMGR       2 Sessions
                                      APPN.AS400   MODE1    Inactive
                                      APPN.BOB     MODE2    4 Sessions
================================================================================
Node           Status         Role                   Description
--------------------------------------------------------------------------------
leia           Inactive       Backup                 Test network
--------------------------------------------------------------------------------
DLC       Port       LS        PU     Type  Status     Description
--------------------------------------------------------------------------------
SDLC0                                   SDLC  Active     SDLC dev 1
          SDLCP0                        SDLC  Inactive   My first por
                     HOST      PU0     SDLC  Inactive   Link to host
--------------------------------------------------------------------------------
PU        LS       NAU  LU       LU type  Status       Description
--------------------------------------------------------------------------------
HOST      SDLC                           Inactive     Link to host
                3    LU1      DISPLAY  Inactive     Freds Display
--------------------------------------------------------------------------------
LU        LU Alias         Machine   Partner LU Mode   Session count
--------------------------------------------------------------------------------
LEIA      L_ALIAS                              Inactive
```

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

status_connectivity

The **status_connectivity** command returns information about the status of all the DLCs, ports, and link stations on the node.

Supplied Parameters

```
[status_connectivity]
```

No parameters are supplied for this command.

Returned Information

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

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

The following example illustrates the information returned for the **status_connectivity** command.

The amount of information returned depends on the width of your display; Communications Server for Linux attempts to determine this using the `COLUMNS` environment variable, and uses a default value of 80 if `COLUMNS` is not set. The *Description* text may be omitted or truncated if there is not enough room to display it.

```
Returned Information for the status_connectivity Command
------------------------------------------------------------------------------
DLC      Port      LS      PU      Type Status    Description
------------------------------------------------------------------------------
TOKEN0                              TR   Active    /dev/tr0
         TRP1                       TR   Inactive  My first por
                   TRL0    PU0      TR   Inactive  Link to host
         TRP2                       TR   Active    My second po
                   TRL1    AS400    TR   Active    Link to AS/4
                   TRL2    PUNAME   TR   Stopping  Link to othe
ETH0                                Eth  Active    Another DLC
         ETHER0                     Eth  Active    My Ethernet
                   ETH0    PU5      Eth  On demand Link for app
         DOWN      PU6              Eth  Inactive  Downstream t
SDLC0                               SDLC  Active     SDLC dev 1
         SDLCP0                     SDLC  Inactive   My first por
                   HOST      PU0    SDLC  Inactive   Link to host
------------------------------------------------------------------------------
```

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

status_dependent_lu

The **status_dependent_lu** command returns information about the status of all dependent LUs on the node.

Supplied Parameters

```
Parameter name              Type        Length      Default
[status_dependent_lu]
pu_name                     character    8          (null string)
lu_type                     constant     ALL
```

Supplied parameters are:

pu_name
: Name of physical unit (PU) used by the dependent LU. When you specify this parameter, status is returned for all dependent LUs that are associated with this PU.

lu_type
: Specifies the type of LUs for which status is to be returned. Possible values are:

ALL Return status for all dependent LUs.

DISPLAY
Return status for all dependent display LUs.

PRINTER
Return status for all dependent printer LUs.

RJE Return status for all dependent LUs used for Remote Job Entry (RJE).

LU6 Return status for all dependent LUs of type 6.2.

OTHER Return status for all dependent LUs that are not used for display, printer, RJE, or dependent LU type 6.2.

Returned Information

The following status information is returned:

- Physical units (PUs) are displayed as either `Inactive` or `SSCP`, depending on whether the PU-SSCP session is active.
- Each logical unit (LU) on the PU is displayed as one of the following:

  Inactive
  Indicates that the session between the LU and the system services control point (the LU-SSCP session) is inactive.

  SSCP Indicates that the session between the primary LU and the secondary LU (the PLU-SLU session) is inactive.

  Active Indicates that both the LU-SSCP and the PLU-SLU sessions are active.

If an LU is in use by an application, Communications Server for Linux displays additional information. Table 5, shows what kind of information is displayed for a given application type.

Table 5. Additional Information by Application Type

Application Type	Information Displayed
Unknown application type	Unknown
LUA application	LUA application server_or_client_hostname
SNA gateway	Downstream LU: dslu_name
FMI application (3270)	3270 display user: user_name Computer: system_name or 3270 printer user: user_name Computer: system_name
TN3270 application	TN3270 address: cfg_ip_address
Dependent LU 6.2	Partner LU: fqplu_name Mode: mode_name

The following example illustrates the information returned for the **status_dependent_lu** command.

The amount of information returned depends on the width of your display; Communications Server for Linux attempts to determine this using the `COLUMNS`

environment variable, and uses a default value of 80 if COLUMNS is not set. The *Description* text may be omitted or truncated if there is not enough room to display it.

```
Returned Information for the status_dependent_lu Command
-------------------------------------------------------------------------------
PU name  Lsname  NAU   LU name  LU type  Status     Description
-------------------------------------------------------------------------------
PU0      TRL0                             Inactive   Link to host
                3     LU1      DISPLAY   Inactive   Freds Display
                4     LU2      PRINTER   Inactive   Fred's printer
                17    DEPLU1   LU62      Inactive   Used by APP1
PU5      ETH0                             SSCP       Link for appl
                10    LU3      RJE       Active     RJE jobs
                                         RJE workstation: WKS1
                                         Computer: george
                11    LU4      OTHER     Inactive   Used for TN32
                12    LU5      DISPLAY   Active     Model 5 displ
                                         3270 display user: liz
                                         Computer: george
                13    DEPLU2   LU62      Active     Used by APP2
                                         Partner LU: APPN.PARTNER
                                         Mode: MODE1
 PU6     DOWN                            Inactive   Downstream to
                99    DSLU99   PRINTER   Inactive   DS for the ot
-------------------------------------------------------------------------------
```

Status about a particular PU is obtained by including the *pu_name* parameter in the command. For example, Communications Server for Linux returns the information shown in the following example, if you enter:

snaadmin status_dependent_lu,pu_name=ETH0

```
Returned Information for a Specified PU on the status_dependent_lu  Command
-------------------------------------------------------------------------------
PU name  Lsname  NAU   LU name  LU type Status    Description
-------------------------------------------------------------------------------
PU5      ETH0                             SSCP      Link for appl
                10    LU3      RJE      Active    RJE jobs
                                         RJE workstation: WKS1
                                         Computer: george
                11    LU4      OTHER    Inactive  Used for TN32
                12    LU5      DISPLAY  Active    Model 5 displ
                                         3270 display user: liz
                                         Computer: george
                13    DEPLU2   LU62     Active    Used by APP2
                                         Partner LU: APPN.PARTNER
                                         Mode: MODE1
-------------------------------------------------------------------------------
```

Status about a particular LU type is obtained by specifying the LU type in the command. You can specify any of the following values:

DISPLAY
: 3270 display LU

PRINTER
: 3270 printer LU

LU62 Dependent LU type 6.2

OTHER Unrestricted type

For example, Communications Server for Linux returns the information shown in the following example, if you enter:

snaadmin status_dependent_lu, lu_type=DISPLAY

```
Returned Information for a Specified LU Type on the status_dependent_lu  Command
------------------------------------------------------------------------------
PU name   Lsname   NAU  LU name  LU type  Status        Description
------------------------------------------------------------------------------
PU0       TRL0                            Inactive      Link to host
                   3    LU1      DISPLAY  Inactive      Freds Display
PU5       ETH0                            SSCP          Link for appl
                   12   LU5      DISPLAY  Active        Model 5 displ
                                          3270 display user: liz
                                          Computer: george
------------------------------------------------------------------------------
```

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

status_dlur

The **status_dlur** command returns information about the status of the node's PUs that use dependent LU requester (DLUR) and their LUs. On a running node, this command also returns information about downstream PUs that use DLUR. Downstream PUs are displayed as `Downstream`. They appear only if they are active.

Supplied Parameters

`[status_dlur]`

No parameters are supplied for this command.

Returned Information

The dependent LU server (DLUS) with which a PU or LU has an active SSCP session appears under the DLUS column. This column displays `Inactive` if no SSCP session is active. If an LU has an active session with a primary LU (a PLU-SLU session), the PLU name is displayed in the DLUS column. The PLU column displays `Inactive` if no PLU-SLU session is active. The following example illustrates the information returned for the **status_dlur** command.

The amount of information returned depends on the width of your display; Communications Server for Linux attempts to determine this using the `COLUMNS`

environment variable, and uses a default value of 80 if COLUMNS is not set. The *Description* text may be omitted or truncated if there is not enough room to display it.

```
Returned Information for the status_dlur Command
-------------------------------------------------------------------------------
DLUR  PU   LU name   DLUS       PLU            Description
-------------------------------------------------------------------------------
DSPU1 (Downstream)   APPN.DLUS
           DLU1      Inactive   Inactive
           DLU2      APPN.DLUS  APPN.PLU2
PU0                  APPN.DLUS                 Host in Naples
           DLU0      APPN.DLUS  APPN.PLU0      Display mod2
PU2                  Inactive                  Host in Athens
           DLU3      Inactive   Inactive       Display mod2
-------------------------------------------------------------------------------
```

The status of a particular PU can be obtained by including the *pu_name* parameter in the command:

snaadmin status_dlur, pu_name=*PUName*

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

status_lu62

The **status_lu62** command returns information about the status of dependent and independent LUs of type 6.2.

Supplied Parameters

```
[status_lu62]
```

No parameters are supplied for this command.

Returned Information

The returned information includes a session count for each combination of local LU, partner LU, and mode that is currently active or has been active since the node was started. The *Machine* parameter displays name of the computer where the transaction program (TP) that is the target for any incoming attaches is running. The following example illustrates the information returned for the **status_lu62** command.

```
Returned Information for the status_lu62 Command
-------------------------------------------------------------------------------
LU name  LU Alias  Machine   Partner LU  Mode       Session count
-------------------------------------------------------------------------------
GEORGE   GEORGE                                     Inactive
FRED     FALIAS    mynode    APPN.AS400  CPSVGMGR   2 Sessions
                             APPN.AS400  MODE1      Inactive
                             APPN.BOB    MODE2      4 Sessions
-------------------------------------------------------------------------------
```

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

status_node

The **status_node** command returns a list of the nodes in the domain and gives their status, configuration role, and description..

Supplied Parameters

```
[status_node]
```

No parameters are supplied for this command.

Returned Information

Status of the node is displayed as one of the following:

- `Inactive`
- `Active`
- `Starting`
- `Stopping`

Configuration role is displayed as one of the following:

- `Master`
- `Backup`
- (blank); indicates that the server is not a master or a backup

The following example illustrates the information returned for the **status_node** command.

The amount of information returned depends on the width of your display; Communications Server for Linux attempts to determine this using the `COLUMNS`

environment variable, and uses a default value of 80 if COLUMNS is not set. The *Description* text may be omitted or truncated if there is not enough room to display it.

```
Returned Information for the status_node
-------------------------------------------------------------------------------
Command Node name     Status         Role       Description
-------------------------------------------------------------------------------
george        Active         Master     Main server
leia          Inactive       Backup     Backup system
queenie       Inactive
-------------------------------------------------------------------------------
```

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

status_remote_node

The **status_remote_node** command returns information about remote nodes and their sessions with the local node (if any).

Parameters

```
[status_remote_node]
```

No parameters are supplied for this command.

Returned Information

The remote node name appears under the Remote System column. Remote nodes may be defined explicitly by defining a partner LU, or they may be determined dynamically when the partner LU establishes a session with a local LU. Explicitly-defined remote nodes always appear in the output whether or not there are any active sessions; dynamic remote nodes appear only if a session is active between the local and remote nodes.

The partner LU name appears under the Partner LU column. The Wildcard column displays Yes if the partner LU name is defined as a wildcard LU name. If the remote LU has an active session with a local LU, the local LU name and mode name are shown. The Session Count column displays Inactive if no session is active.

The following example illustrates the information returned for the **status_dlur** command.

The amount of information returned depends on the width of your display; Communications Server for Linux attempts to determine this using the COLUMNS environment variable, and uses a default value of 80 if COLUMNS is not set. The *Description* text may be omitted or truncated if there is not enough room to display it.

```
Returned Information for the status_remote_node Command
-------------------------------------------------------------------------------
Remote System       Partner LU          Wildcard  Local LU  Mode      Session Count
-------------------------------------------------------------------------------
APPN.ACENODE
APPN.BARTLOCN
                    APPN.BARTLOCN                                     Inactive
APPN.REMNODE
                    APN.FRED            Yes                           Inactive
                    APPN.PART                                         Inactive
                    APPN.PART2                                        Inactive
                    APPN.REMNODE                                      Inactive
                    APPN.TCPIP                                        Inactive
                    APPN.WILD           Yes                           Inactive
APPN.SOS1
APPN.ZAMBIA
-------------------------------------------------------------------------------
```

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

stop_dlc

The **stop_dlc** command requests Communications Server for Linux to stop a DLC. This command also stops any active ports and link stations that use the DLC.

This command must be issued to a running node.

Supplied Parameters

```
Parameter name          Type            Length    Default
[stop_dlc]
stop_type               constant                  ORDERLY_STOP
dlc_name                character       8
```

Supplied parameters are:

stop_type
Type of stop process required. Possible values are:

ORDERLY_STOP
Communications Server for Linux performs cleanup operations before stopping the DLC.

IMMEDIATE_STOP
Communications Server for Linux immediately stops the DLC.

dlc_name
Name of DLC to be stopped. This must match the name of a defined DLC.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_DLC
The *dlc_name* parameter did not match the name of a defined DLC.

UNRECOGNIZED_DEACT_TYPE
The *stop_type* parameter was not set to a valid value.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc

STOP_DLC_PENDING
The specified DLC is already being stopped.

Other Conditions

If the command does not execute because other conditions exist, Communications Server for Linux returns the following parameters:

primary_rc
CANCELLED

secondary_rc

NO_SECONDARY_RC
The *stop_type* parameter specified an orderly stop, but the DLC was then stopped by a second command specifying an immediate stop, or by a failure condition.

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

stop_internal_pu

The **stop_internal_pu** command requests DLUR to initiate SSCP-PU session deactivation for a previously defined local PU that is served by DLUR.

This command must be issued to a running node.

Supplied Parameters

Parameter name	Type	Length	Default
[stop_internal_pu]			
pu_name	character	8	
stop_type	constant		ORDERLY_STOP

Supplied parameters are:

pu_name
Name of the internal PU for which the SSCP-PU session will be deactivated. This name is a type-A character string starting with a letter.

stop_type
Specifies how to stop the PU. Possible values are:

ORDERLY_STOP
Deactivate all underlying PLU-SLU and SSCP-LU sessions before deactivating the SSCP-PU session.

IMMEDIATE_STOP
Deactivate the SSCP-PU session immediately.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc

INVALID_STOP_TYPE
The *stop_type* parameter was not set to a valid value.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

PU_NOT_DEFINED
The supplied PU name did not match the name of a defined internal PU.

PU_ALREADY_DEACTIVATING
The PU is already being deactivated.

PU_NOT_ACTIVE
The PU is not active.

Function Not Supported

If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns the following parameters:

primary_rc

FUNCTION_NOT_SUPPORTED
The node does not support DLUR; this support is defined by the *dlur_support* parameter on the **define_node** command.

secondary_rc
(This parameter is not used.)

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

stop_ls

The **stop_ls** command stops an active LS. Alternatively, the command can be issued for an inactive LS to specify that the LS cannot be automatically activated by Communications Server for Linux when required, or cannot be activated by the remote system; if both of these are disabled, the LS can be activated only by issuing **start_ls**.

This command must be issued to a running node.

Supplied Parameters

```
Parameter name       Type        Length   Default
[stop_ls]
stop_type            constant             ORDERLY_STOP
ls_name              character   8
disable              constant             NO
```

Supplied parameters are:

stop_type
Type of stop processing required. Possible values are:

ORDERLY_STOP
Communications Server for Linux performs cleanup operations before stopping the LS.

IMMEDIATE_STOP
Communications Server for Linux stops the LS immediately.

ls_name
Name of LS to be stopped.

disable Specifies the action to be taken for the LS.

To stop an active LS and return to the default settings for auto-activation and remote activation, set this parameter to NO.

To specify that an inactive LS cannot be activated by Communications Server for Linux, or cannot be activated by the remote system, specify one or both of the following values (combined with a + character):

AUTO_ACT
The LS cannot be automatically activated by Communications Server for Linux.

REMOTE_ACT
The LS cannot be activated by the remote system. This value does not alter the defined value of *disable_remote_act* in the LS definition; when the LS is next started and stopped, it will return to the defined setting.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

LINK_NOT_DEFD
The *ls_name* parameter did not match the name of a defined LS.

UNRECOGNIZED_DEACT_TYPE
The *stop_type* parameter was not set to a valid value.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc

LINK_DEACT_IN_PROGRESS
The specified LS is already being deactivated.

Other Conditions

If the command does not execute because other conditions exist, Communications Server for Linux returns the following parameters:

primary_rc
CANCELLED

secondary_rc

NO_SECONDARY_RC
The *stop_type* parameter specified an orderly stop, but the LS was then stopped by a second command specifying an immediate stop, or by a failure condition.

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

---

stop_port

The **stop_port** command stops a port. This command also stops any active link stations that are using the port.

This command must be issued to a running node.

Supplied Parameters

Parameter name	Type	Length	Default
[stop_port]			
stop_type	constant		ORDERLY_STOP
port_name	character	8	

Supplied parameters are:

stop_type
Type of stop processing required. Possible values are:

ORDERLY_STOP
Communications Server for Linux performs cleanup operations before stopping the port.

IMMEDIATE_STOP
Communications Server for Linux immediately stops the port.

port_name
Name of the port to be stopped.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

If the command does not execute because of a parameter error, Communications Server for Linux returns the following parameters:

primary_rc
PARAMETER_CHECK

secondary_rc
Possible values are:

INVALID_PORT_NAME
The *port_name* parameter did not match the name of a defined port.

UNRECOGNIZED_DEACT_TYPE
The *stop_type* parameter was not set to a valid value.

State Check

If the command does not execute because of a state error, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc

STOP_PORT_PENDING
The specified port is already being deactivated.

Other Conditions

If the command does not execute because other conditions exit, Communications Server for Linux returns the following parameters:

primary_rc
CANCELLED

secondary_rc

NO_SECONDARY_RC
The *stop_type* parameter specified an orderly stop, but the port was then stopped by a second command, specifying an immediate stop, or by a failure condition.

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

term_node

The **term_node** command stops a node with a specified urgency and also stops all connectivity resources associated with the node.

This command must be issued to a running node.

Supplied Parameters

Parameter name	Type	Length	Default
[term_node]			
stop_type	constant		SHUTDOWN

Supplied parameters are:

stop_type
Specifies how Communications Server for Linux stops the node. Possible values are:

ABORT Stop immediately without attempting any cleanup processing. This value should be used only in serious error conditions because it may cause problems for other programs that are using the node's resources.

SHUTDOWN
Deactivate all link stations associated with the node before stopping.

QUIESCE
Indicate to the APPN network that the node is quiesced, reset session limits on all modes, unbind all sessions for the node's LUs, and then stop as for SHUTDOWN. For a network node, any ISR sessions active through this node will be terminated.

QUIESCE_ISR
Same functions as QUIESCE, except that the node waits for all intermediate sessions to end. This value applies only to network nodes.

DEACT_CLEAN
Same functions as QUIESCE, except that session limits are not reset and RTP connections are allowed to terminate gracefully before the link stations are deactivated.

Returned Parameters

No parameters are returned by Communications Server for Linux when this command executes successfully.

Error Return Codes

If the command cannot be executed, Communications Server for Linux returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for unsuccessful execution.

Parameter Check

No parameter errors occur for this command.

State Check

No specific state errors, other than those listed in Appendix A, "Common Return Codes from snaadmin Commands," on page 589, occur for this command.

Other Conditions

Appendix A, "Common Return Codes from snaadmin Commands," on page 589, lists combinations of primary and secondary return codes that are common to all commands.

Appendix A. Common Return Codes from snaadmin Commands

This section describes the primary and secondary return code values that are common to all **snaadmin** commands. Return codes that are specific to a particular command are described in the individual command descriptions.

Communications Subsystem Not Active

If the command does not execute because a required component is not active, Communications Server for Linux returns the following parameters:

primary_rc
COMM_SUBSYSTEM_ABENDED

secondary_rc
Possible values are:

LOCAL_ABENDED
The Communications Server for Linux software has stopped.

TARGET_ABENDED
The target node has stopped, or the communication path to it has failed.

primary_rc

COMM_SUBSYSTEM_NOT_LOADED
The Communications Server for Linux software is not active.

secondary_rc
(This parameter is not used.)

primary_rc

NODE_NOT_STARTED
The target node has not been started. This command must be issued to an active node.

secondary_rc
(This parameter is not used.)

primary_rc

NODE_STOPPING
The target node is in the process of stopping. This command must be issued to an active node.

secondary_rc
(This parameter is not used.)

Function Not Supported

If the command does not execute because the node's configuration does not support it, Communications Server for Linux returns one of the following parameters:

primary_rc
INVALID_VERB

secondary_rc
(This parameter is not used.)

primary_rc
FUNCTION_NOT_SUPPORTED

secondary_rc
(This parameter is not used.)

Parameter Check

There are no common parameter check return codes. Parameter check return codes that are specific to a particular command are described in the individual command descriptions.

State Check

If the command does not execute because of a state check, Communications Server for Linux returns the following parameters:

primary_rc
STATE_CHECK

secondary_rc
Possible values are:

CANT_MODIFY_VISIBILITY
You have attempted to define a resource with a name that is reserved for use internally by Communications Server for Linux. Please choose a different name.

FILE_LOCKED
Another administration program or NOF application has locked the configuration file. Wait for the other application to complete its processing and try again.

If this condition persists, you may be able to clear the lock by running the command **verifysna -R**.

FILE_UNAVAILABLE
The connection to the target configuration file has been lost.

INVALID_VERSION
The Communications Server for Linux version number in the configuration file header does not match the version number of the Communications Server for Linux software you are using. Check that you have the correct file.

NOT_AUTHORIZED
You do not have permission to issue this administration command because your login ID is not a member of the SNA administrators group sna. You can issue **query_*** or **status_*** commands to view information about Communications Server for Linux resources, but you cannot modify, start, or stop resources.

System Error

If the command does not execute because of a system error, Communications Server for Linux returns the following parameters:

primary_rc

UNEXPECTED_SYSTEM_ERROR
An operating system call failed while the command was being processed.

secondary_rc
In this case, the secondary return code is the return code from the operating system call. For the meaning of this return code, check the returned value in the **errno.h** file on the computer where the error occurred.

If the command was issued to change the target configuration (such as **define**_* or **delete**_*), or to perform an action (such as **start**_*), issue the appropriate **query**_* command to determine whether the change or action was successful. If this error occurs while processing a **define**_* or **delete**_* command containing subrecords , the change may be incomplete.

Appendix B. Configuration Files

This appendix describes:

- Initial definition of Communications Server for Linux node and domain resources
- Format of configuration files
- Changes made to the node and domain resources by the Motif program
- File input to the **snaadmin** program

Initial Configuration Files

Configuration records for a node's resources are included in the node configuration file. When you start the Communications Server for Linux software, the configuration file **/etc/opt/ibm/sna/sna_node.cfg** is used as the initial definition of the node's configuration. Communications Server for Linux uses the information in this file to define the resources available when the node is started and to start any resources that you have specified as being initially active.

Configuration records for domain resources are included in the domain configuration file, instead of in individual node configuration files. For information about the distinction between domain resources and node resources, refer to the *Communications Server for Linux Administration Guide*.

When you start the Communications Server for Linux software on the master server, the domain configuration file **/etc/opt/ibm/sna/sna_domn.cfg** is used as the initial definition of Communications Server for Linux domain resources.

If a file cannot be opened or if it contains information that is not valid, the Communications Server for Linux node will not start. For more information about starting Communications Server for Linux, refer to Communications Server for Linux Administration Guide.

Configuration File Format

A Communications Server for Linux configuration file is an ASCII text file with information stored in readable text format. You can set up or check your configuration using a standard ASCII text editor.

Although you can modify configuration files using a text editor, you can do this only when the Communications Server for Linux software is not running. You are not recommended to modify the files in this way except when setting up the initial configuration (before starting the Communications Server for Linux software). To modify the configuration while the Communications Server for Linux software is running, use the command-line administration program or the Motif administration program. If you need to modify a node's configuration file using a text editor, the Communications Server for Linux software must not be running on the node or on the server for that node.If you need to modify the domain configuration file using a text editor, you must first stop the Communications Server for Linux software on all servers, modify the file on the master server, and then restart the Communications Server for Linux software on the master server before restarting it on any other servers.

Note: Both Communications Server for Linux configuration files are regenerated by the owning server when a configuration command is issued or when configuration is changed using the Motif interface. If you have changed the file using a text editor while the Communications Server for Linux software is running, these conditions will overwrite your changes to the file, and the sequence of fields in the file may be changed.

A configuration file consists of a [define_node_config_file] or [define_domain_config_file] header record followed by a series of [define_*] and [set_*] administration records. Each administration record contains the parameters for a Communications Server for Linux administration command. Header records and administration records are used as follows:

- The header record contains information such as the Communications Server for Linux version number.
- The [define_*] administration records define the available resources: a local node and its resources (node resources), or resources not associated with a specific node (domain resources).
- The [set_*] administration records set parameters that determine how Communications Server for Linux operates, such as the locations of diagnostics files and the types of diagnostics information to record.

A node configuration file consists of a [define_node_config_file] header record, a [define_node] record defining the node, and a series of [define_*] and [set_*] records defining the node's resources. The domain configuration file consists of a [define_domain_config_file] header record and a series of [define_*] records and [set_*] records defining the domain resources.

The other types of administration commands (such as **start_***, **stop_***, and **delete_***) are not used in a configuration file; those commands are used only when administering a running Communications Server for Linux system.

For information about the order of these records within the file, see "Record Ordering in a Configuration File."

Record Ordering in a Configuration File

In a node configuration file, the first record is the [define_node_config_file] header record, which defines the Communications Server for Linux version number and the file's revision level. The header record must be followed by a [define_node] record, and then by [define_*] and [set_*] records for all the resources associated with the node. The [define_node_config_file] record is set up automatically by Communications Server for Linux when the configuration file is created; you cannot access this record using the **snaadmin** program, and must not attempt to modify it when editing the file.

In the domain configuration file, the first record is the [define_domain_config_file] header record, which defines the Communications Server for Linux version number and the file's revision level (and optionally includes a comment string describing the contents of the file). The header record must be followed by [define_*] records for domain resources. There is no restriction on the ordering of domain resource records.

Record Format

Each record is defined in the following format:

```
[command_name]
parameter_name   =       value
parameter_name   =       value
       .
       .
       .
parameter_name   =       value
```

The *command_name* must be enclosed in square brackets. It is followed by a series of parameter entries, each on a separate line. A backslash character (\) at the end of a line indicates that the entry continues on the next line.

All the parameters associated with a particular record must be listed after the *command_name* for that record, and before the *command_name* for the next record in the file. However, the order of individual parameters within a record is not important (except where this is indicated in the command descriptions). Also, Communications Server for Linux provides defaults for many parameters, so you do not need to specify every parameter explicitly. For more information, see "Parameter Syntax Used for Administration Commands" on page 4.

The following example shows one way the [define_lu_0_to_3] record can be specified. For full details of the parameters associated with this command, see "define_lu_0_to_3" on page 90. Because the *priority* parameter is not included, Communications Server for Linux uses the default value of MEDIUM. The optional parameters *description* and *pool_name* are also not included.

```
[define_lu_0_to_3]
lu_name  =  LU$01
nau_address = 1
pu_name = PU2
lu_model = 3270_DISPLAY_MODEL_2
```

Subrecord Format

Some configuration records include data whose format can vary between instances of that record type. For example, the [define_cos] record includes a variable number of node rows and TG rows. To handle this variability, the variable data is specified in optional subrecords. This means that a record consists of a series of parameters common to all instances of that record type, followed by subrecords containing the variable data.

A record that contains one or more subrecords is defined as follows:

```
[command_name]
parameter_name = value
               .
               .
parameter_name = value

{subrecord_name}
parameter_name = value
               .
               .
parameter_name = value

{subrecord_name}
parameter_name = value
               .
               .
parameter_name = value
```

The *subrecord_name* must be enclosed in braces. It is followed by a series of parameter entries associated with this subrecord, each on a separate line.

All the parameters associated with the *command_name* (and not with a subrecord) must be listed after the *command_name* and before the first *subrecord_name*; all the parameters associated with a particular *subrecord_name* must be listed after that *subrecord_name* and before the next *subrecord_name*, if any, or the next *command_name*. However, the order of individual parameters within a subrecord is not important. For more information, see "Parameter Syntax Used for Administration Commands" on page 4.

Changes Made to the Configuration Files by the Motif Administration Program

When you use the Motif administration program to configure parameters, the Motif program updates the node and domain configuration files. The entries in the configuration file may differ from what you entered on the Motif screens in the following ways:

- If you enter a name on a Motif screen using fewer characters (or fewer hexadecimal bytes) than allowed for that name, Communications Server for Linux pads the name with blank characters (or expands the hexadecimal value) to make the length equal to the maximum length allowed (or full hexadecimal width) for that name. For example, if you enter `Node1` for the *node_name* parameter (which allows 128 characters) when you are defining a node, Communications Server for Linux pads `Node1` with 123 blank characters so that the value in the node configuration record has the maximum length allowed for this parameter.
- If you enter hexadecimal digits `A`, `B`, `C`, `D`, `E`, and `F` on a Motif screen, Communications Server for Linux changes them to `a`, `b`, `c`, `d`, `e`, and `f` in the configuration file.
- If you do not enter a value on a Motif screen for a parameter that defaults to a null string, Communications Server for Linux adds a null string for that parameter's value in the configuration file.
- Communications Server for Linux substitutes some command names. For example, if you define an adjacent LEN node with a Motif screen, Communications Server for Linux substitutes a `[define_directory_entry]` record in the configuration file. For more information about the relationship between defining an adjacent LEN node and defining a directory entry, see "define_directory_entry" on page 46.

File Input to the snaadmin Program

The command-line administration program **snaadmin**, can take its input from a text file instead of directly from the command line. The file format used for a **snaadmin** input file is the same as the Communications Server for Linux configuration file format; the information in this section applies to **snaadmin** as well as to configuration files used when starting the Communications Server for Linux software.

The only differences between the format of configuration files and **snaadmin** input files are:

- A configuration file used at startup can include only records corresponding to **define_*** and **set_*** commands; the **snaadmin** input file can include records corresponding to all the different types of administration commands (**define_***, **set_***, **start_***, **stop_***, **query_***, and **delete_***). The records for the additional commands are included in the **snaadmin** file using the same format as for

[define_*] and [set_*] records. For information about the usage of these commands, see Chapter 1, "Introduction," on page 1.

- The configuration file contains the complete configuration of a Communications Server for Linux node or of Communications Server for Linux domain resources; the **snaadmin** input file can contain either complete information or partial information (to modify or query an existing configuration).
- The [define_node_config_file] and [define_domain_config_file] header records are not required in the **snaadmin** input file.

Appendix C. Environment Variables

This appendix provides an alphabetical listing of all the environment variables that are used by Communications Server for Linux programs. It includes a brief summary of how Communications Server for Linux uses each variable, and provides a cross-reference to additional information provided elsewhere in the Communications Server for Linux documentation set.

Most of these environment variables are specific to Communications Server for Linux programs. However, a small number are standard Linux environment variables that may be used by other programs on your computer; you may need to modify your settings of these variables for use with other programs as well as Communications Server for Linux programs.

Environment Variables That Affect All Functions

LANG

The setting of the LANG environment variable determines the language used for online help and message catalogs supplied by Communications Server for Linux.

PATH

Communications Server for Linux uses the PATH environment variable to specify where executable programs are stored on the Linux computer.

The programs are stored in the directory **/opt/ibm/sna/bin**. If you add this directory to the definition of the PATH environment variable in your **.login** or **.profile** file, the programs are located automatically.

Alternatively, you can specify the directory name when you run the program, as in the following example:

/opt/ibm/sna/bin/snaadmin init_node

The sample command lines shown in the Communications Server for Linux manuals assume that you have added the directory to your PATH environment variable, and do not include the directory names.

LD_PRELOAD

If you have built an application using the Communications Server for Linux APIs with Communications Server for Linux version 6.0, you may need to use LD_PRELOAD to ensure that the application works correctly with LiS Streams. Set LD_PRELOAD to the value **/usr/lib/libpLiS.so** (for a 32–bit application) or **/usr/lib64/libpLiS.so** (for a 64–bit application) before starting the application.

If you need to run an existing 32–bit application on a 64–bit system, you must export the 32–bit version of LD_PRELOAD only for that application; other programs may fail if they are run with this setting.

Environment Variables That Affect APPC and CPI-C Communications

APPCLLU

The Communications Server for Linux CPI-C library uses APPCLLU to specify the name of the local APPC LU used by a CPI-C application. The local LU alias to be used for a CPI-C application can be configured using the **define_cpic_side_info** command. The environment variable APPCLLU overrides that alias.

If you choose to set APPCLLU, use an LU alias value (1–8 characters), not a fully qualified LU name (which consists of a network name of 1–8 characters, followed by a period, followed by a local LU name of 1–8 characters).

If you do not set APPCLLU before starting the application, the program uses a default local LU.

For more details, refer to the information about local LUs for CPI-C applications in *Communications Server for Linux CPI-C Programmer's Guide*.

APPCTPN

The Communications Server for Linux CPI-C library uses APPCTPN to specify the local TP name used by a CPI-C application. If you do not set APPCTPN before starting the application, the program uses the default value CPIC_DEFAULT_TPNAME.

For more details, refer to the information about TP names for CPI-C applications in *Communications Server for Linux CPI-C Programmer's Guide*.

LD_LIBRARY_PATH

Java™ CPI-C applications use LD_LIBRARY_PATH to specify a directory containing runtime libraries used by a CPI-C application.

For more details, refer to the information about compiling and linking Java CPI-C applications in *Communications Server for Linux CPI-C Programmer's Guide*.

CLASSPATH

Java CPI-C applications use CLASSPATH to specify a directory containing Java Classes used by a Java CPI-C application.

For more details, refer to the information about compiling and linking Java CPI-C applications in *Communications Server for Linux CPI-C Programmer's Guide*.

LD_PRELOAD

Java CPI-C applications use LD_PRELOAD to ensure that Java CPI-C works correctly with LiS Streams.

For more details, refer to the information about compiling and linking Java CPI-C applications in *Communications Server for Linux CPI-C Programmer's Guide*.

Environment Variables That Affect the CSV API

SNATBLG

The Communications Server for Linux CSV library uses SNATBLG to specify the user-defined translation table file (Table G) that is used for ASCII-EBCDIC conversions.

If you are running a CSV application that uses the CONVERT verb for Table G conversions, set SNATBLG to the full path name of the translation table file. Otherwise, you do not need to set SNATBLG.

For more information, see the description of the CONVERT verb in *Communications Server for Linux CSV Programmer's Guide*.

Environment Variables That Affect the Command-Line Administration Program

COLUMNS

Communications Server for Linux uses COLUMNS to control the display of information returned on the **status_*** administration commands.

The amount of information that can be returned depends on the width of your display; Communications Server for Linux attempts to determine this using the COLUMNS environment variable, and uses a default value of 80 if COLUMNS is not set. The *Description* text may be omitted or truncated if there is not enough room to display it.

For more details, see "status_all" on page 570.

Environment Variables That Affect Tracing

SNATRC

Communications Server for Linux uses SNATRC to control API tracing on applications using the Communications Server for Linux APIs.

For more details, refer to the information about controlling tracing on user-space components in *Communications Server for Linux Diagnostics Guide*.

SNACTL

The Communications Server for Linux API libraries use SNACTL to suppress control of tracing from within applications.

If API tracing is active (specified using the SNATRC environment variable), an application can use the CSV DEFINE_TRACE call or the HLLAPI Set Session Parameters call to turn tracing on and off while the application is running. You can prevent these calls from taking effect by setting SNACTL to any non-null string. If SNACTL is not set, or is null, the calls will operate normally.

For more details, refer to the information about controlling tracing on user-space components in *Communications Server for Linux Diagnostics Guide*.

SNATRACESIZE

The Communications Server for Linux API libraries use SNATRACESIZE to specify the maximum size of API trace files.

If API tracing is set up to use two files (specified using the SNATRC environment variable), tracing switches between the two files each time the file size reaches the limit specified by SNATRACESIZE. If SNATRACESIZE is not set, Communications Server for Linux uses a default file size limit of 1,000,000 bytes.

For more details, refer to the information about controlling tracing on user-space components in *Communications Server for Linux Diagnostics Guide*.

SNATRCRESET

The Communications Server for Linux API libraries use SNATRCRESET to specify whether an API trace file is reset when an application first writes to it.

Normally, the file is reset (and its existing contents are discarded) when an application writes its first trace message to the file. If you are tracing two or more applications to the same file, or if you want to trace two or more runs of the same application to the same file, you need to prevent the file from being reset. To do this, set SNATRCRESET to NO. If SNATRCRESET is not set, or is set to YES, Communications Server for Linux resets the file when an application first writes to it.

For more details, refer to the information about controlling tracing on user-space components in *Communications Server for Linux Diagnostics Guide*.

SNATRUNC

The Communications Server for Linux API libraries use SNATRUNC to specify the maximum length of data stored for each trace message that is written to API trace files.

Set SNATRUNC to a decimal number specifying the maximum number of bytes to be traced from each message. Excess bytes are ignored and are not written to the trace file. If SNATRUNC is not set, Communications Server for Linux traces each message in full.

For more details, refer to the information about controlling tracing on user-space components in *Communications Server for Linux Diagnostics Guide*.

Appendix D. Notices

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Trademarks

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

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

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

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

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

Intel is a trademark of Intel Corporation.

Linux is a trademark of Linus Torvalds.

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

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

Bibliography

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

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

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

Communications Server for Linux Version 6.2.3Publications

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Systems Network Architecture (SNA) Publications

The following books contain information about SNA networks:

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

Host Configuration Publications

The following books contain information about host configuration:

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

z/OS Communications Server Publications

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

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

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

TCP/IP Publications

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

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

X.25 Publications

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

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

APPC Publications

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

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

Programming Publications

The following books contain information about programming:

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

Other IBM Networking Publications

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

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

Index

E

F

H

I

K

L

M

N

P

Q

R

S

T

U

Communicating Your Comments to IBM

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

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

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

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

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

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

Make sure to include the following in your note:

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

Program Number:

Printed in USA

SC31-6770-02