IBM Communications Server for Data Center Deployment on Linux
Version 7.1

Node Operator Facility Programmer's Guide

SC31-6778-05

Note

Before using this information and the product it supports, be sure to read the general information under Appendix D, “特記事項,” on page 671.

セブンス・エディション (2021 年 1 月)

本書は、 IBM Communications Server for Data Center Deployment on Linux、バージョン 7.1、プログラム番号 5725-H32 、および新しい版またはテクニカル・ニュースレターで明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

お客様の地域にサービスを提供する IBM 担当員または IBM 営業所の資料をご注文ください。 資料は、以下の住所に保管されていません。

IBM はご自身のコメントを 本書の裏側には、読者のコメントの形式が記載されています。 フォームが削除されている場合は、コメントを以下のアドレスに送信できます。

- インターナショナル・ビジネス・マシン
- 部門 CGMD
- おや ボックス 12195
- リサーチ・トライアングル・パーク(ノースカロライナ州)
- 27709-2195
- 米国 .

コメントを電子的に送信したい場合は、以下のいずれかの方法を使用します。

- IBMLink: RALVM17 の CIBMORCF
- IBM Mail: IBMMAIL における USIB2HPD
- インターネット : USIB2HPD@vnet.ibm.com
- FAX : 1-800-227-5088

IBM に情報を送信する場合、お客様に対していかなる義務も負うことのない適切な方法で、情報を使用または配布するための非独占的な権利を IBM に付与します。

Contents

Tables

Figures

About this book

IBM Communications Server for Data Center Deployment on Linux NOF Programmer's Guide contains the information required to develop C-language application programs that use the Node Operator Facility (NOF) API to manage IBM Communications Server for Data Center Deployment on Linux resources. IBM Communications Server for Data Center Deployment on Linux, program product number 5725-H32, is an IBM®software product that enables a computer running Linux to exchange information with other nodes on an SNA network.

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

CS Linux
CS Linux operates on the following:

- 64-bit AMD64/Intel EM64T workstations running Linux (x86_64)
- IBM Power computers running Linux (ppc64le)

CS Linux for IBM Z
CS Linux for IBM Z operates on IBM Z mainframes running Linux for IBM Z (s390x).

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

This book applies to Version 7.1 of CS Linux.

本書の対象読者

本書は、 CS Linux を備えたシステム用のシステム・ネットワーク体系 (SNA) トランザクション・プログラムを作成する、経験豊富な C プログラマーを対象にしています。

本書は、 CS Linux を使用するシステム管理者およびアプリケーション・プログラマーを対象としています。

システム管理者
システム管理者は CS Linux をインストールし、ネットワーク接続用にシステムを構成し、システムを保守します。 これらは、 CS Linux が稼働するハードウェア、および Linux オペレーティング・システムに精通している必要があります。 また、システムが接続されているネットワークについての知識も豊富であり、一般的に SNA の概念を理解している必要があります。

アプリケーション・プログラマー
アプリケーション・プログラマーは、 CS Linux プログラミング・インターフェースを 使用して、 SNA ネットワークを介してデータを送受信するトランザクション・プログラムおよびアプリケーション・プログラムを設計し、コーディングします。 これらは、 SNA、トランザクションまたはアプリケーション・プログラムが通信するリモート・プログラム、および AIX または Linux オペレーティング・システムのプログラミング環境およびオペレーティング環境に精通している必要があります。

アプリケーション・プログラムの作成についての詳しい情報は、各 API のマニュアルに記載されています。 CS Linux 資料に関する追加情報については、「参考文献」を参照してください。

本書の使用方法

このセクションでは、本書で説明する情報の編成方法について説明します。

本書の構成

本書の編成は以下のとおりです。

- 1 ページの『第 1 章 Introduction to the NOF API』では、 CS Linux NOF API の概要と、その API が提供する機能の概要を示します。
- 19 ページの『第 2 章 Writing NOF Applications』には、 NOF アプリケーションの作成時にプログラマーが必要とする一般情報、およびアプリケーションのコンパイルとリンクに関する情報が含まれています
- 37 ページの『第 3 章 NOF API verbs』は、パラメーターおよび戻りコードを含む、各 NOF verb の詳細な説明を提供します。
- 603 ページの『第 4 章 NOF Indications』は、 NOF アプリケーションが受信できるように登録できる各指示について、詳細な説明を提供します。
- 657 ページの『付録 A 戻りコードの値』は、 NOF インターフェースに 指定可能なすべての戻りコードを番号順にリストして、その意味を示します。
- 665 ページの『付録 B 共通戻りコード』は、すべての NOF verb に共通する戻りコードに関する情報を提供します。

Typographic conventions

Table 1 on page xxxiv shows the typographic styles used in this document.

Table 1. Typographic Conventions

Special Element	Sample of Typography
Document title	*IBM Communications Server for Data Center Deployment on Linux NOF Programmer's Guide*
File or path name	`sna.err`
Directory name	`/var/opt/ibm/sna`
Header file	`nof_c.h`
Program or application	`snaadmin`
Command	`define_local_lu`; `cd`
General reference to all verbs of a particular type	DEFINE_* (indicates all of the NOF API verbs that define resources)
Option or flag	`-I`
Parameter	*opcode*
Literal value or selection that the user can enter (including default values)	`255`
Constant	`AP_MODE_READ_ONLY`
Return value	`AP_INVALID_FORMAT`; `0`
Variable representing a supplied value	*a.b.c.d*
Environment variable	`LD_RUN_PATH`
Programming verb	CONNECT_NODE
User input	`snaadmin status_dependent_lu,pu_name=ETH0`
Function, call, or entry point	`ioctl`
Data structure	`NOF_CALLBACK`
Hexadecimal value	`0x20`

グラフィックの規則

このシンボルは、 AIX または Linux オペレーティング・システムのみに適用されるテキスト・セクションの開始を示すために使用されます。 これは、 Linux サーバー、および AIX 、 Linux、 Linux for Power または Linux for IBM Z で稼働する IBM Remote API クライアントに適用されます。

このシンボルは、 Windows 上の IBM Remote API Client に適用されるテキストのセクションの開始を示すために使用されます。

このシンボルは、オペレーティング・システム固有のテキストのセクションの終わりを示します。 このシンボルに続く情報は、オペレーティング・システムに関係なく適用されます。

詳細情報の参照先

CS Linux ライブラリー内の他のブックについては、「参考文献」を参照してください。また、 SNA および Linux ワークステーションに関連するトピックに関する追加情報を記載した資料も参照してください。

Chapter 1. Introduction to the NOF API

This chapter provides an introduction to the CS Linux NOF API. It includes the following information:

- Purpose of the NOF API
- Client/server operation
- NOF verbs and indications

For information about the CS Linux components and resources accessed by the NOF API, see *IBM Communications Server for Data Center Deployment on Linux Quick Beginnings*.

Purpose of the NOF API

The CS Linux NOF API provides access to a standard set of commands, called NOF verbs, that can be used to administer the CS Linux system from within an application program. These verbs enable you to define and delete resources, specify CS Linux parameters such as diagnostics levels and file names, start and stop defined resources, query the definition or current status of resources, and manage which servers on the CS Linux LAN can act as backup controllers if the controlling configuration file server is not available.

In a client/server system, you can use any NOF verbs in an application running on a server. Applications running on Remote API Clients can use NOF verbs to query configuration or status information, but cannot use other verbs to modify the configuration or to start or stop resources.

The NOF verbs provide the same functions as commands issued to the command-line administration program `snaadmin`, or as records in a CS Linux configuration file. For example, the NOF verb DEFINE_LOCAL_LU is equivalent both to a `define_local_lu` command issued to the `snaadmin` program, and to a `define_local_lu` record in a configuration file; all three of them perform the same function, which is to specify the parameters of a CS Linux local APPC LU.

You can use the Motif administration program `xsnaadmin` to perform the same function as a NOF verb or an administration command (for example, to define a local APPC LU). However, this does not provide access to the full range of parameters included in some NOF verbs. For more information about using the Motif administration program, refer to the *IBM Communications Server for Data Center Deployment on Linux Administration Guide*.

You can issue NOF verbs to any of the following targets:

- A running CS Linux node - to manage its resources or to monitor its operation
- A server where the node is not running - to query the stored configuration or to modify it for use when the node is next started
- The CS Linux domain as a whole - to define, modify, or query the configuration of domain resources (resources used to support particular user programs, such as CPI-C side information entries, which are not associated with a particular node).
- The CS Linux invokable TP data file - to define information that CS Linux needs to start invokable (target) TPs, or to define other information relating to a TP (such as the level of security required to access the TP).

The NOF API enables you to do the following:

- Develop your own application programs to manage the CS Linux system
- Develop application programs that use the other CS Linux APIs so that they can also manage their own resources (for example, an APPC application can check that the communications link to its partner TP is active before attempting to allocate a conversation or can define the remote LU where its partner TP is located).

ノード構成ファイル

各 CS Linux ノードの構成情報は、ノードが実行されているコンピューター上のテキスト・ファイルに保持されます。 このファイルには、ノードのリソースに関する情報が含まれ、 CS Linux の開始時にどのリソースがアクティブになるかが指定されます。 ノードを開始すると、ファイルは、使用可能なリソースの初期定義を提供します。その後、 NOF API または CS Linux 管理ツールを使用して、実行中のノードのリソースを要件変更として変更することができます。

複数の構成ファイルをセットアップし、異なる時間で使用するために異なる CS Linux 構成を保管し、これらのファイルを選択して、 CS Linux ソフトウェアの開始時に使用することができます。

アタン ネットワークでの構成は動的プロセスです。 CS Linux ソフトウェアの実行中に、必要に応じてリソースを追加、削除、または変更することができます。 構成ファイルは、使用可能なリソースの初期定義を提供し、現行の定義を保管して、ノードを再始動する必要があるときに再び使用できるようにします。 ただし、 CS Linux ソフトウェアを開始する前に構成全体を定義する必要はありません。

Domain configuration file

Configuration information for CS Linux domain resources is held in a single text file on the controller server. You can set up multiple domain configuration files, to store different CS Linux configurations for use at different times, and select which of these files to use when starting the CS Linux software on the controller server.

Configuration in an APPN network is a dynamic process; you can add, delete, or modify resources as necessary while the system is running. The domain configuration file provides an initial definition of the available domain resources and enables you to store the current definition so that you can use it again when you need to restart the system, but it is not necessary to define the entire domain configuration before starting the CS Linux software or to restart the software when you make changes.

Invokable TP data file

Information that CS Linux needs to start invokable (target) TPs is held in the file `/etc/sna/sna_tps` (AIX) or `/etc/opt/ibm/sna/sna_tps` (Linux). This file can also provide other information (such as the level of security required to access the TP). The invokable TP data file resides on the computer where the TPs run.

CS Linux コンポーネント

CS Linux は、 APPN ノードを実装して、 SNA ネットワーク上の他のノードと通信します。これは、 APPC および CPI-C 機能に対する論理装置 (LU) 6.2 サポート、および 3270 および LUA 通信用の LU 0、 1、 2、および 3 のサポートを提供します。

CS Linux は、その構成に応じて、 LEN ノード、エンド・ネットワーク・ノード、ネットワーク・ノード、または分岐ネットワーク・ノードのいずれかとして作動することができます。 特定の機能は、 APPN アーキテクチャーで定義されているように、特定のノード・タイプでのみサポートされます。 これらの相違点は、本書で必要とされる箇所で示されています。相違は示されていないため、すべてのノード・タイプに情報が

3 ページの図 1 には、 CS Linux のコンポーネントと、それらがどのように連動するかを示します。

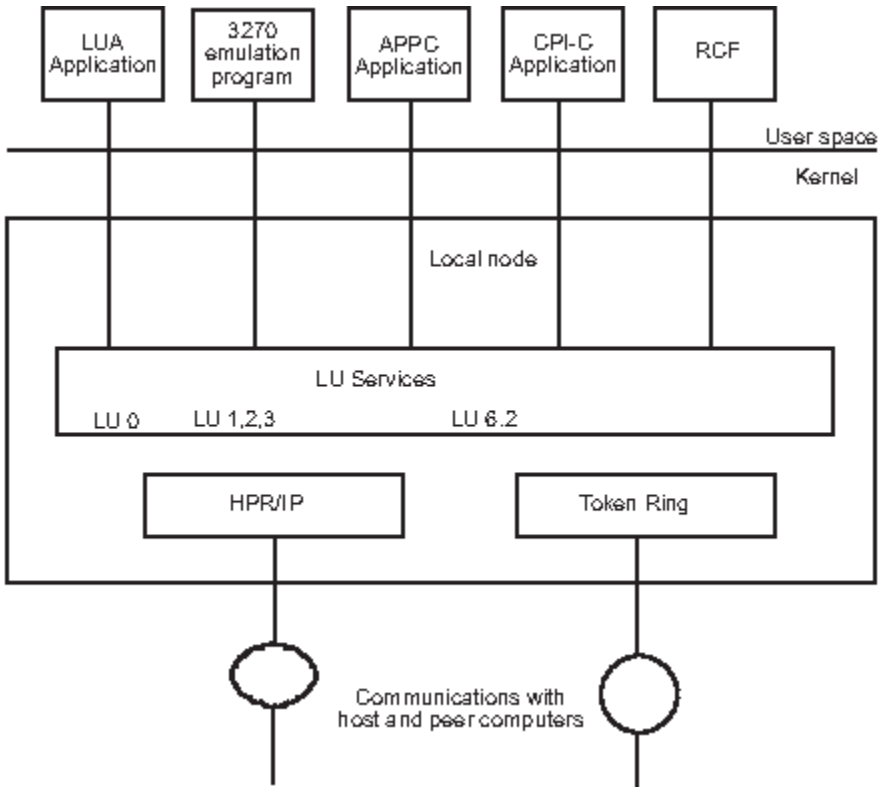

図 *1. CS Linux* コンポーネント

ローカル・ノードは、関連する接続リソース (DLC、ポート、および LSs) を含めて、 Linux システムのカーネル内に CS Linux コンポーネントとして実装されます。

APPC トランザクション・プログラム、 CPI-C アプリケーション、 LUA アプリケーション、およびリモート・コマンド機能 (RCF) は、ユーザー・スペース・プログラムです。 CS Linux は、並行して実行される複数の APPC TP、 CPI-C アプリケーション、および LUA アプリケーションをサポートします。

Client/Server operation

The computers on the CS Linux network are of two types: servers and clients. A server contains a CS Linux node and its associated connectivity components; a client does not contain these connectivity components, but accesses them on the server by means of the network. Servers are Linux computers; clients can be running AIX, Linux, or Windows. (An AIX or Linux computer can be either a server or a client, but not both; you cannot install both the server and the client on the same computer.) Servers and clients communicate across the network using Berkeley Software Distribution (BSD) Sockets.

Each CS Linux network, referred to as a domain, is identified by a domain name. This name is specified during the installation of each CS Linux computer (server or client), so that all computers in a single CS Linux network have the same domain name. To install two separate CS Linux domains on the same physical network, you simply use two different domain names to identify the domain in which each computer belongs. A single CS Linux domain can correspond to a TCP/IP subnet, can be part of a TCP/IP subnet (so that there are two or more separate CS Linux domains in the same subnet), or can span multiple subnets.

Each server maintains information about its own node configuration in a node configuration file. You can use the CS Linux administration tools or the NOF API to examine the node's configuration. This can be done either from this server or from any other computer in the domain, as long as the SNA software is running (whether or not the node is started). You can also use the CS Linux administration tools or the NOF API on this server or on any other server to modify the node's configuration or to start or stop resources on the node.

Information about the configuration of domain resources for the complete CS Linux network is held in a domain configuration file. If you have more than one server on the network, CS Linux ensures that this information is consistent across all servers.

Controller server and backup servers

If you are using CS Linux with all programs on one computer or on a network that contains only one server, you do not need to read this section.

At any time, one server on the network, known as the controller server, holds the controlling copy of the CS Linux domain configuration file. You can define other servers on the network to be backup servers; the domain configuration file is copied to backup servers (either when they are started or when the controlling copy is changed) so that all backup servers hold a copy of the latest information.

If the controller server fails or if the SNA software on that computer is stopped, a backup server takes over as the controller. The domain configuration file on this server is used as the controlling copy and is copied to other servers as necessary. When the controller server is restarted, it receives a copy of the domain configuration from the backup server currently acting as controller and then takes over as the controller.

In general, define at least one backup server in addition to the controller server. Any remaining servers can be defined as additional backup servers or they can be left as peer servers. A peer server obtains configuration information from the controller server as required but cannot act as a backup server.

If at any time the controller server and all backup servers are inactive, a node on a peer server can still operate, and you can still change the node's configuration. However, you cannot access the domain configuration file and therefore cannot access the configuration of domain resources (as opposed to node resources). This means that you will not be able to allocate CPI-C conversations using symbolic destination names defined in the configuration file.

There is one situation in which CS Linux cannot maintain a consistent configuration of domain resources across the network; it is your responsibility to maintain the configuration in this case. This situation occurs when the network is split by a network failure into two noncommunicating domains, each containing one or more backup servers. In this situation, there will be an acting controller server in each domain, which will hold any changes made to the domain configuration file in that domain but will be unaware of any changes made in the other domain. When the network connection is re-established, the domain configuration file from the original controller server (or from the highest backup server available in either of the two domains if the controller is inactive at this point) will become the domain configuration file across the network; this will overwrite any changes made to the domain configuration file in the other domain while the network was split. Because of this, do not attempt to make any changes to the domain configuration file in either of the two domains while the network connection is broken. Changes can be made to the configuration of individual nodes.

CS Linux stores information about the controller server and backup servers in the file `sna.net`, known as the SNA network data file. The controlling copy of this file is stored on the controller server; any changes made to it are automatically copied to all other servers, in the same way that changes to the domain configuration file are copied to backup servers. You cannot edit the contents of the file directly; instead, CS Linux provides NOF verbs to access the file.

For more information about the SNA network data file, refer to the *IBM Communications Server for Data Center Deployment on Linux Administration Command Reference*.

CS Linux clients can run either on AIX, Linux or Windows systems or within containers on these systems (or AIX WPARs). CS Linux Servers cannot run within containers since they have Unix kernel dependencies.

AIX or Linux clients

Clients may be running inside Containers on the AIX or Linux Operating System on inside AIX WPARs.

A client computer does not contain any configuration file or the SNA network data file; it holds only the information it needs to access servers on the CS Linux network and relies on a server to provide the necessary configuration information.

The SNA network information required is held in the file `/etc/sna/sna_clnt.net` (AIX) or `/etc/opt/ibm/sna/sna_clnt.net` (Linux). For more information about this file, refer to the *IBM Communications Server for Data Center Deployment on Linux Administration Command Reference*.

On a client, you can use the NOF API to query configuration, initialize or activate sessions, and manage local logging and tracing options. You cannot modify a node's configuration, or start or stop resources on the node.

Windows クライアント

クライアントは、 Windows オペレーティング・システムのコンテナー内で実行されている場合があります

Windows Server ® 2012 、 Windows Server ® 2012 、 Windows Server ® 2012、 Windows Server ® 2012、 Windows Server ®、 Windows Server ® 、 Windows Server ® 、 Windows Server ® 、 Windows Server ® 、 Windows Server ® 、 Windows Server ® Windows クライアントが必要とする構成情報は、 Windows レジストリーを介して管理されます。

Windows レジストリーの詳細について、および Windows クライアントの管理については、「*IBM Communications Server for Linux* 管理ガイド上のデータ・センター・デプロイメント」を参照してください。

クライアントでは、 NOF API を使用して、構成の照会、セッションの初期化またはアクティブ化、およびローカル・ロギングおよびトレースのオプションの 管理を行うことができます。 ノードの構成を変更したり、ノード上のリソースの開始または停止を行うことはできません。

NOF verbs to manage specific CS Linux functions

The following sections list the NOF verbs that are relevant to particular CS Linux functions. For more information about individual verbs, see Chapter 3, “NOF API verbs,” on page 37.

Managing the Target (Node or File) for NOF Verbs

A NOF verb can be issued to a node, to the domain configuration file, or to the SNA network data file. To access the target node or file, use one of the following verbs:

- OPEN_FILE
- CONNECT_NODE

When you issue the verbs shown above to access the target, you are initially restricted to issuing verbs that query the configuration; you cannot issue verbs to modify it. If the NOF application is running on a server (not on a client), you can obtain write access to the target node or file so that you can issue verbs that modify the configuration. Use the following verb:

- SET_PROCESSING_MODE

To register for indications when the target configuration changes, use the following verb:

- REGISTER_INDICATION_SINK

To unregister when indications are no longer required, use the following verb:

- UNREGISTER_INDICATION_SINK

To release the target node or file when you have finished issuing NOF verbs, use one of the following verbs:

- DISCONNECT_NODE, CLOSE_FILE

You can issue OPEN_FILE, CONNECT_NODE, DISCONNECT_NODE, and CLOSE_FILE verbs, and NOF QUERY verbs, from an application running on a client, as well as from an application running on a server. You cannot issue any other NOF verbs from the client.

Getting started

The first step is to define the CS Linux node that runs on each computer, and its communications links to other computers. To define these components, use the following verbs:

- DEFINE_NODE
- DEFINE_DLC, DEFINE_PORT, DEFINE_LS

After defining these components, activate them to establish the link to the remote system. (DLCs, ports, and LSs can be defined to be "initially active" using the DEFINE_* verbs described previously, so that they are started automatically when the node is started; in this case, it is not necessary to start them manually.) To activate components, use the following verbs:

- INIT_NODE
- START_DLC, START_PORT, START_LS

The components must be started in the order shown because each component relies on the one before it.

To stop these components when access to the remote system is no longer required, use the following verbs:

- STOP_LS, STOP_PORT, STOP_DLC

To obtain information about the configuration or current status of these components, use the following verbs:

- QUERY_NODE
- QUERY_DLC, QUERY_PORT, QUERY_LS

To obtain information about the usage of an LS or port, use the following verb:

- QUERY_STATISTICS

To delete connectivity components when they are no longer required, use the following verbs:

- DELETE_DLC, DELETE_PORT, DELETE_LS

If you are communicating with many nodes on the same shared-access transport facility (SATF), you can set up a connection network (CN) to represent these nodes, instead of having to define explicit LSs to each node. CNs cannot be used if the local node is a LEN node.

To set up the CN, you first define a DLC and port to access each of the nodes on the SATF.

You then define a CN that includes all these ports; you do not need to define any LSs because a dynamic LS to the CN will be set up as required. To define the CN, or to add ports to an existing CN, use the following verb:

- DEFINE_CN

To obtain information about defined CNs, or about the ports on a CN, use the following verbs:

- QUERY_CN, QUERY_CN_PORT

To delete a CN when it is no longer required, or to remove ports from a CN without deleting the CN, use the following verb:

- DELETE_CN

To stop the node, which deactivates all resources associated with it, use the following verb:

- TERM_NODE

To define default parameters used by the node, or to query the definition of these parameters, use the following verbs:

- DEFINE_DEFAULTS, QUERY_DEFAULTS

To query the options and limits permitted by your CS Linux license for the node, use the following verb:

- QUERY_NODE_LIMITS

3270 通信

CS Linux ユーザーがホスト・システムと通信するために 3270 エミュレーションを使用する場合は、ホストへの通信リンクを定義する必要があります。 詳しくは、 6 ページの『Getting started』を参照してください。 ホストへの LS の定義には、 3270 エミュレーションに必要な LU を所有するローカル PU の名前が含まれていなければならず、 *solicit_sscp_sessions* パラメーターが **類人猿**に設定されている必要があります。

次に、 3270 エミュレーションに使用できる LU を定義する必要があります。 これを行うには、以下の verb を使用します。

- クラスター・アドレス 3、 1 から 3 までの値の値を定義する

LU の構成または現在の状況に関する情報を取得するには、次の verb を使用します。

- 照会 LU_0_TO_3

LU を所有している PU に関する情報を入手するには、次の verb を使用します。

- クエリー・プ

LU が必要でなくなったときに LU を削除するには、次の verb を使用します。

- 削除 0 から 3、削除された値の最大値 (3)

LU プール (各ユーザー・セッションごとに LU を明示的に定義するのではなく、必要に応じてユーザー・セッションに割り当てることができる LU のグループ) を提供する場合は、以下の verb を使用してプールを定義するか、定義に関する情報を入手するか、またはプールを削除するか、不要になったときに LU を除去します。

- 定義 LU_POOL、 QUERY_LU_POOL、 DELETE_LU_POOL

LUA communications

If applications running on CS Linux will be using LUA to communicate with host programs, you need to define the communications link to the host. For more information, see “Getting started” on page 6. The definition of the LS to the host must include the name of a local PU to own the LUs, and must have the *solicit_sscp_sessions* parameter set to AP_YES.

You then need to define LUs that can be used for LUA. To define the LUs, use the following verbs:

- DEFINE_LU_0_TO_3 to define an individual LU or DEFINE_LU_0_TO_3_RANGE to define multiple LUs with a single verb

To delete LUs when they are no longer required, use the following verbs:

- DELETE_LU_0_TO_3 to delete an individual LU or DELETE_LU_0_TO_3_RANGE to delete multiple LUs with a single verb

To obtain information about the configuration or current status of LUs, use the following verb:

- QUERY_LU_0_TO_3

To obtain information about the PU that owns an LU, use the following verb:

- QUERY_PU

If you want to provide LU pools (groups of LUs that can be assigned to applications as required, rather than having LUs explicitly defined for each application), use the following verbs to define a pool, to obtain information about the definition, or to delete a pool or remove LUs from it when no longer required:

- DEFINE_LU_POOL, QUERY_LU_POOL, DELETE_LU_POOL

If applications running on CS Linux will be using LUA to communicate with applications on downstream computers, you need to define the LUs on the downstream computer and map these to the LUs on the CS Linux node. To define the downstream LUs, use the following verbs:

- DEFINE_DOWNSTREAM_LU, DEFINE_DOWNSTREAM_LU_RANGE, DEFINE_DSPU_TEMPLATE

To obtain information about the configuration or current status of downstream LUs or about the downstream PU that serves them, use the following verbs:

- QUERY_DOWNSTREAM_LU, QUERY_DOWNSTREAM_PU, QUERY_DSPU_TEMPLATE

To delete downstream LUs when they are no longer required, use the following verbs:

- DELETE_DOWNSTREAM_LU, DELETE_DOWNSTREAM_LU_RANGE, DELETE_DSPU_TEMPLATE

APPC communications

If applications running on CS Linux will be using APPC to communicate with applications running on host or peer computers, you need to define LUs that can be used for APPC.

APPC configuration in an APPN network is much simpler than in a pre-APPN SNA network. Many of the required components, and the interactions between them, can be defined or determined dynamically when sessions are started and do not need to be specified explicitly in the initial configuration.

Each node includes a default APPC local LU (the "control point LU"). An APPC application can use this LU, or you can define additional LUs so that different applications can use different LUs. To define the LUs, use the following verb:

- DEFINE_LOCAL_LU

To obtain information about the configuration or current status of LUs, including the control point LU, use the following verb:

- QUERY_LOCAL_LU

Because APPN can locate a partner LU dynamically when a local application needs to start a session to it, normally you do not need to define partner LUs. However, you may need to define partner LUs if you need to enforce the use of particular APPC features such as conversation security. To define a partner LU, use the following verb:

- DEFINE_PARTNER_LU

To obtain information about the current status of a partner LU or about its definition if it was explicitly defined, use the following verbs:

- QUERY_PARTNER_LU, QUERY_PARTNER_LU_DEFINITION

If the local application communicates with its partner using one of the standard SNA-defined modes, you do not need to define a mode. However, you may want to define additional modes for applications that have particular requirements not covered by the standard modes. To define a mode, use the following verb:

- DEFINE_MODE

To define or query the default mode, which specifies parameters that will be used for any unrecognized mode name, use the following verbs:

- DEFINE_DEFAULTS, QUERY_DEFAULTS

The class of service (COS) used for a mode is normally one of the standard SNA-defined classes of service. However, the node can be configured to support mapping each mode to a specific COS (the *mode_to_cos_map_supp* parameter on the DEFINE_NODE verb). In this case, you may want to define additional COSs for applications that have particular requirements not covered by the standard COSs. To define a COS, use the following verb:

- DEFINE_COS

To specify the default COS to which any unrecognized modes will be mapped, use the following verb:

- DEFINE_MODE

To obtain information about the definition or current usage of a mode, about the COS used by a mode, or about the definition of a COS, use the following verbs:

- QUERY_MODE_DEFINITION, QUERY_MODE, QUERY_MODE_TO_COS_MAPPING

- QUERY_COS, QUERY_COS_NODE_ROW, QUERY_COS_TG_ROW

If the local and partner LUs use session-level security, you need to define the password used to establish a session between the local LU and partner LU. To define the password, check the current definition, or delete the password when it is no longer required, use the following verbs:

- DEFINE_LU_LU_PASSWORD, QUERY_LU_LU_PASSWORD, DELETE_LU_LU_PASSWORD

To delete local LUs, partner LUs, modes, or COSs when they are no longer required, use the following verbs:

- DELETE_LOCAL_LU, DELETE_PARTNER_LU
- DELETE_MODE, DELETE_COS

CS Linux negotiates session limits with the partner LU automatically when sessions are established. If you need to manage session limits between a local LU and its partner LU explicitly, use the following verbs:

- INITIALIZE_SESSION_LIMIT, CHANGE_SESSION_LIMIT, RESET_SESSION_LIMIT

To manage individual sessions and conversations, use the following verbs:

- QUERY_SESSION, QUERY_ISR_SESSION, QUERY_CONVERSATION
- ACTIVATE_SESSION, DEACTIVATE_SESSION, DEACTIVATE_CONV_GROUP

Normally you do not need to define CS Linux invokable TPs if they are operator-started. If a TP is to be automatically started by CS Linux when a remote TP allocates a conversation to it, if it is to be operator-started and a broadcast queued TP (which means that incoming conversation requests can be routed dynamically to the TP wherever it is running), or if it is to be operator-started and requires a specific Receive_Allocate timeout value, you need to specify it in the CS Linux invokable TP data file. For more information about this file, refer to the *IBM Communications Server for Data Center Deployment on Linux Administration Guide*.

In addition, if a TP (either operator-started or auto-started) needs to be restricted to particular values for conversation security, confirm synchronization, or conversation type (mapped or basic), or if you need to limit the number of instances of the TP that can be running at any time, you need to define the TP. Use the following verb:

- DEFINE_TP

To obtain information about the definition of a TP, about its current usage, or about currently active invokable TPs, use the following verbs:

- QUERY_TP_DEFINITION, QUERY_TP, QUERY_AVAILABLE_TP

To delete a defined TP when it is no longer required, use the following verb:

- DELETE_TP

If the invokable TP requires conversation-level security, you need to define user IDs and passwords that remote TPs can use to access CS Linux TPs. To define user IDs and passwords, check the current definitions, or delete user IDs and passwords when they are no longer required, use the following verbs:

- DEFINE_USERID_PASSWORD, QUERY_USERID_PASSWORD, DELETE_USERID_PASSWORD

To restrict the use of the TP to a specific list of authorized user IDs, check the current list of authorized user IDs, or delete a list of user IDs when it is no longer required, use the following verbs:

- DEFINE_SECURITY_ACCESS_LIST, QUERY_SECURITY_ACCESS_LIST, DELETE_SECURITY_ACCESS_LIST

CPI-C communications

CPI-C applications use the same resources as APPC applications; the information in "APPC communications" on page 8, applies to CPI-C as well as to APPC.

In addition, you can set up side information entries for use by CPI-C applications; each entry defines a particular partner application and the information required to access it. The local CPI-C application can

then identify its partner application simply by the name of a side information entry, instead of having to specify explicit partner LU and TP names, mode name, and conversation security requirements. To define side information entries, check the current definitions, or delete entries when they are no longer required, use the following verbs:

- DEFINE_CPIC_SIDE_INFO, QUERY_CPIC_SIDE_INFO, DELETE_CPIC_SIDE_INFO

HPR RTP 接続の管理

RTP 接続のセットアップ時に使用されるチューニング・パラメーターを定義するには、次の verb を使用します。

- デフォルトのチューニング

RTP 接続をセットアップするときに使用するために現在定義されているチューニング・パラメーターを確認するか、現在アクティブな RTP 接続の詳細を確認するには、以下の verb を使用します。

- 照会 RTP_チューニング、 QUERY_RTP_CONNECTION

SNA ゲートウェイの管理

ノードが SNA ゲートウェイ (DEFINE_NODE verb の *pu_conc__* サポート パラメーター) をサポートしている場合は、ダウンストリーム・コンピューター上のタイプ 0 から 3 の LU を、 CS Linux ノードで定義された LU を使用してホスト・システムと通信できるようにするには、まず以下を定義する必要があります。

- DLC、ポート、および LS を CS Linux からダウンストリーム・コンピューターに提供します。 これらのコンポーネントの定義については、 6 ページの『Getting started』を参照してください LS は以下のパラメーターを使用して定義する必要があります。

```
solicit_sscp_sessions   =  NO
dspu_services    =  PU_CONCENTRATION

dspu_name =  the name of the PU serving the LUs on the downstream computer
pu_name  =  all zeros
```

- ホストとの通信のために、 CS Linux ノード上の 1 つ以上のタイプ 0 から 3 の LU (およびオプションでこれらの LU を含む LU プール) 。 LU および LU プールの定義については、 7 ページの『3270 通信』を参照してください。

次に、ダウンストリーム・コンピューター上の LU を定義し、これらを CS Linux ノード上の LU にマップします。 ダウンストリーム LU を定義するには、以下の verb を使用します。

- 定義が STREAMSTREAM_LU、 DEFINE_DOWNSTREAM_LU_RANGE

ダウンストリーム LU の構成または現在の状況、またはそれらにサービスを提供するダウンストリーム PU に関する情報を取得するには、以下の verb を使用します。

- QUERY_DOWNSTREAM_LU、 QUERY_DOWNSTREAM_PU

ダウンストリーム LU が不要になったときに削除するには、以下の verb を使用します。

- 削除が簡素化されました。削除された値の範囲が削除されます。

Managing DLUR

If the node supports DLUR (the *dlur_support* parameter on the DEFINE_NODE verb), and LUs on the CS Linux node will be using DLUR to communicate with host systems, you need to define the PU on the local CS Linux node that owns these LUs. This is not the same as defining a PU for LUs that communicate directly with the host (which is done using the DEFINE_LS verb).

To define the PU, use the following verb:

- DEFINE_INTERNAL_PU

To obtain information about the PU, use the following verb:

- QUERY_PU

To define and manage the LUs associated with this PU, see “3270 通信” on page 7 or “LUA communications” on page 7, earlier in this section.

To start the PU (to request an ACTPU from the host) in order to use the LUs or to stop it when applications are no longer using the LUs, use the following verbs:

- START_INTERNAL_PU, STOP_INTERNAL_PU

To delete the PU when it is no longer required, use the following verb:

- DELETE_INTERNAL_PU

If the local node is a network node, and LUs on downstream PUs will be using DLUR to communicate with host systems, you need to define the communications link to the downstream PU, as described in “Getting started” on page 6. The LS definition must specify that the local node provides DLUR services to the downstream PU.

You do not need to define the downstream PUs; CS Linux will obtain the necessary information dynamically when communications links are established. To obtain information about downstream PUs and LUs currently using DLUR, use the following verbs:

- QUERY_DOWNSTREAM_PU, QUERY_DOWNSTREAM_LU

To set up defaults to simplify DLUR configuration and reduce the information required on other DLUR verbs, use the following verb:

- DEFINE_DLUR_DEFAULTS

To obtain information about PUs and LUs currently using DLUR (either on the local node or on downstream PUs), or about the DLUS nodes they are using, use the following verbs:

- QUERY_DLUR_PU, QUERY_DLUR_LU, QUERY_DLUS

Managing TN server

If TN3270 users will be using the TN server feature on a CS Linux node to communicate with host systems, you need to define the communications link to the host. For more information, see “Getting started” on page 6. The definition of the LS to the host must include the name of a local PU to own the 3270 LUs and must have the *solicit_sscp_sessions* parameter set to AP_YES.

You then need to define LUs that can be used for 3270 emulation and optionally group these LUs into LU pools. For more information about defining LUs and pools, see “3270 通信” on page 7.

To define parameters that apply to all TN Server users, use the following verb:

- 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, you need to configure details of how to access this server. In addition, if the client users are permitted to use the TN3270 Express Logon feature, so that their security certificate authorization replaces the standard user ID and password normally used for TN3270 security, you need to configure the host Digital Certificate Access Server (DCAS) used to manage this feature. Use the following verbs:

- DEFINE_TN3270_SSL_LDAP
- DEFINE_TN3270_EXPRESS_LOGON

To define the TN3270 users that can access TN server and assign them to CS Linux 3270 LUs, use the following verb:

- DEFINE_TN3270_ACCESS

To define the association between TN3270 display and printer LUs, so that a TN3270E client can connect to the printer LU that is associated with a display LU without knowing the name of the printer LU, use the following verb:

- DEFINE_TN3270_ASSOCIATION

To obtain information about the configuration of TN Server and TN3270 users, use the following verbs:

- QUERY_TN3270_ACCESS_DEF, QUERY_TN3270_ASSOCIATION, QUERY_TN3270_DEFAULTS, QUERY_TN3270_SSL_LDAP, QUERY_TN3270_EXPRESS_LOGON

To delete TN3270 users so that they can no longer use TN server for 3270 emulation, or to delete LU association information, use the following verbs:

- DELETE_TN3270_ACCESS, DELETE_TN3270_ASSOCIATION

TN リディレクターの管理

Telnet ユーザーが CS Linux ノードの TN リダイレクター機能を使用してホスト・システムと通信する場合は、これらのユーザーを定義し、それらがホストにアクセスする方法を定義する必要があります。

TN リダイレクターにアクセスできる TN3270 ユーザーを定義するには、次の verb を使用します。

- 定義済みのリダイレクト

Secure Sockets Layer (SSL) クライアント認証を使用しており、外部 LDAP サーバー上の証明書失効リストに対してクライアントを検査する場合は、このサーバーへのアクセス方法の詳細を構成する必要があります。 さらに、クライアント・ユーザーが TN3270 高速ログオン機能の使用を許可されている場合、それらのセキュリティー証明書の許可が、 TN3270 セキュリティーに通常使用される標準のユーザー ID とパスワードを置き換えるようにするには、この機能を管理するために使用するホスト・デジタル証明書アクセス・サーバー (DCAS) を構成する必要があります。 以下の動詞を使用します。

- 定義済みの LDAP の最大値は 1 つです

TN リダイレクターおよび TN リダイレクター・ユーザーの構成に関する情報を取得するには、以下の verb を使用します。

- QUERY_TN_REDIRECT_DEF 、 QUERY_TN3270_SSL_LDAP

TN リダイレクターを使用してホストにアクセスできなくなるように TN リダイレクター・ユーザーを削除するには、以下の verb を使用します。

- 宛先変更の削除

SNA 管理サービス機能の管理

CS Linux で実行中のアプリケーションが MS API を使用してリモート MS アプリケーションと通信する場合は、この明示的にリソースを定義する必要はありません。これは、ノードが、必要に応じて適切なリモート・アプリケーションを見つけるためです。 ただし、使用する特定のリモート・アプリケーションを指定したい場合は、リソースを明示的に定義することができます。

NMVT レベルのアプリケーションが使用するデフォルト PU を指定する (特定のホストで NetView プログラムにアクセスするために) には、次の verb を使用します。

- 定義のデフォルトの T_PU

MDS レベルのアプリケーションで使用するフォーカル・ポイント・アプリケーションを指定するには (リモート・フォーカル・ポイント・アプリケーションで管理対象のノードを判別する代わりに) 、次の verb を使用します。

- 定義ポイント・フォーカル・ポイント

現在使用中のフォーカル・ポイントに関する情報を入手するため、または以前に定義されたフォーカル・ポイントを削除するには、次の verb を使用します。

- 照会フォーカル・ポイント、 DELETE_FOCAL_POINT

MS 機能を使用して、アクティブ・アプリケーション (NMVT レベルまたは MDS レベル) に関する情報を取得するには、以下の verb を使用します。

- 照会 NMVT_APPLICATION、 QUERY_MDS_APPLICATION

MDS レベル・アプリケーションからの未処理の要求に関する情報を取得したり、以前の要求に関する統計情報を取得したりするには、以下の verb を使用します。

- 照会アクティブ・トランザクション、照会の照会の統計

ホスト NetView プログラムからの CS Linux システムへのアクセスの管理

ホスト NetView コンソールでオペレーターを使用可能にして、サービス・ポイント・コマンド機能 (SPCF) または UNIX コマンド機能 (UCF) を使用して CS Linux コンピューター上でコマンドを発行できるようにするには、これらのオペレーターに対するアクセス許可を定義する必要があります。

これらの許可を定義し、 NetView オペレーターが SPCF または UCF またはその両方にアクセスできるようにするには、次の verb を使用します。

- 定義済み CF_ACCESS

現在定義されている許可を確認するには、次の verb を使用します

- クエリー _RCF_ACCESS

オペレーターが SPCF または UCF を使用しないようにするには、次の verb を使用します。

- アクセス権の削除 (_R)

1 つの機能へのアクセスを除去するが、他の機能は使用できない場合は、次の verb を使用

- 定義済み CF_ACCESS

Managing diagnostics settings

The CS Linux default setting for log messages is to log problem and exception messages but not audit messages, and to use local logging (messages from each server are written to a file on that server) rather than central logging (messages from all servers are sent to a central log file on the controller server). Succinct logging is used (that is, logging of header parameters and message text, but not full details of cause and action for each message). The error log file, used for problem and exception messages, is `/var/opt/ibm/sna/sna.err`; the audit log file, used for audit messages if these are enabled, is `/var/opt/ibm/sna/sna.aud`. Each of these files is backed up and reset when the file size reaches 10 megabytes. The default settings for succinct logging, exception and audit logging, file names, and file sizes can all be overridden using NOF verbs, as described in the following information.

The verbs to manage central logging and global logging options apply to clients as well as to servers. However, other diagnostics settings on Windows clients are controlled by options in the Windows Registry, and not by NOF verbs. For more information, refer to the *IBM Communications Server for Data Center Deployment on Linux Administration Guide*.

CS Linux also maintains a usage log file `/var/opt/ibm/sna/sna.usage`, which is used to record information about the current and peak usage of CS Linux resources. This file is backed up and reset in the same way as the error and audit log files, and the file name and file size can be specified in the same way.

To specify whether central logging is enabled, use the following verb:

- SET_CENTRAL_LOGGING

To specify whether exception messages or audit messages or both are logged, or to specify whether succinct logging or full logging is to be used, either to establish global default settings for all servers or to override the defaults for a particular server, use the following verbs:

- SET_GLOBAL_LOG_TYPE, SET_LOG_TYPE

To change the file names or directories used for log messages or to change the size at which files are backed up and reset, use the following verb:

- SET_LOG_FILE

To check which server is currently defined as the central logger or to check whether central logging is enabled, use the following verbs:

- QUERY_CENTRAL_LOGGER, QUERY_CENTRAL_LOGGING

To check the types of messages being recorded or to check whether succinct logging or full logging is being used, either globally or on a particular server, use the following verbs:

- QUERY_GLOBAL_LOG_TYPE, QUERY_LOG_TYPE

To check the file, file size, or directory being used for a particular log type, use the following verb:

- QUERY_LOG_FILE

If you want to activate tracing to diagnose problems with connectivity components on a particular CS Linux node or to deactivate it after collecting the required data, use the following verbs:

- ADD_DLC_TRACE, REMOVE_DLC_TRACE

If you want to activate tracing to diagnose problems with other CS Linux kernel components or to deactivate it after collecting the required data, use the following verb:

- SET_TRACE_TYPE

If you want to activate tracing to diagnose problems with communications between clients and servers across the CS Linux LAN or to deactivate it after collecting the required data, use the following verb:

- SET_CS_TRACE

If you want to activate tracing to diagnose problems with the CS Linux TN server feature or to deactivate it after collecting the required data, use the following verbsverb:

- SET_TN_SERVER_TRACE

The default files used for trace data are as follows:

- `/var/opt/ibm/sna/sna1.trc` and `/var/opt/ibm/sna/sna2.trc` for tracing on a particular computer
- `/var/opt/ibm/sna/snacs1.trc` and `/var/opt/ibm/sna/snacs2.trc` for LAN tracing
- `/var/opt/ibm/sna/snatnsv1.trc` and `/var/opt/ibm/sna/snatnsv2.trc` for TN server tracing

If you want to use different files or directories for either of these trace types or to send all tracing of a particular type to one file instead of two files, use the following verb:

- SET_TRACE_FILE

To check the current settings for a particular trace type or to check the files used for a particular trace type, use the following verbs:

- QUERY_DLC_TRACE, QUERY_TRACE_TYPE, QUERY_CS_TRACE, QUERY_TN_SERVER_TRACE, QUERY_TRACE_FILE

ディレクトリー項目の管理

ローカル・ノードが LEN ノードである場合は、 CS Linux が通信する隣接ノードと、これらのノードに関連付けられている LU を識別するために、ローカル・ノードのディレクトリー内にエントリーをセットアップする必要があります。 特定のノードに類似した名前の LU の範囲が含まれている場合は、その範囲内のすべての LU が指定されたノード上にあることを示すために、ディレクトリー内にワイルドカード・エントリーを設定できます。

ノードおよびそれに関連する LU を定義するには、次の verb を使用します。

- 定義された値の定義ノード

データベース内の特定のノードまたは LU エントリーに関する情報を取得するには (ただし、この verb を使用してワイルドカード・エントリーに関する情報を戻すことはできません) 、以下の verb を使用します。

- QUERY_DIRECTORY_ENTRY

データベース内の特定の LU エントリーまたはワイルドカード・エントリーに関する情報を取得するには、次の verb を使用します。

- クエリー・ディレクトリー・ルー

ディレクトリー項目に関する統計情報を取得するには、次の verb を使用します。

- クエリー・ディレクトリー統計

ノードおよびそのノードに関連付けられた LU を削除したり、ノード項目から LU を削除したりするには、次の verb を使用します。

- 区切り文字の削除のノード・ノード

ローカル・ノードが LEN ノードと通信しているエンド・ノードまたはネットワーク・ノードである場合、またはローカル・ノードが LEN ノードにサービスを提供しているネットワーク・ノードである場合は、LEN ノードおよびその LU のディレクトリー項目をセットアップする必要があります。これは上記の動詞を使用して行われます。 他のノード・タイプとの通信の場合、ノードはこれらのリソースを必要に応じて動的に見つけるため、ディレクトリー項目をセットアップする必要はありません (また、これらのリソースを再び使用できるようにディレクトリーに追加します) 。

ただし、特定のノードまたは LU の項目をセットアップして、ローカル・ノードがそれらのノードを検索することなく、これらのリソースと通信できるようにすることもできます。 特定のノードまたは LU のエントリーをセットアップすると、通常の APPN リソース・ロケーション・プロセスがオーバーライドされるため、定義が正しくない場合には、このノードまたはネットワーク内の他のノードで問題を発生させることができます。 定義が正しいことを確認していない限り、他のノードでリソースに明示的な項目を定義しないでください。

LU の範囲について、個々のノード、 LU、またはワイルドカード項目を定義するには、次の verb を使用します。

- 定義ディレクトリー・エントリー

個々のノード、 LU、またはワイルドカード・エントリーをディレクトリーから削除するには、次の verb を使用します。

- ディレクトリー・エントリーの削除

前に説明した verb を使用して明示的に定義されたディレクトリー項目だけを削除します (これらの項目は、 QUERY_DIRECTORY_ENTRY verb で **ホーム** の項目タイプを戻します) 。 キャッシュされた項目 (ネットワーク検索の結果として動的にセットアップされている項目) を削除する場合には、この verb を使用しないでください。

ネットワーク・トポロジーの照会

隣接ネットワーク・ノードに関する情報 (ネットワーク・ノード上で) を取得するには、次の verb を使用します。

- 照会の従属 NN

隣接ネットワーク・ノードに対する TG に関する情報を取得するには、次の verb を使用します。

- クエリー・ローカル・トポロジー

ネットワーク・ノードおよびネットワーク内の仮想ルーティング・ノード (VRN) 、またはこれらのノードへの TG に関する情報 (ネットワーク・ノード上) を取得するには、以下の verb を使用します。

- 照会 NN_TOPOLOGY_NODE、 QUERY_NN_TOPOLOGY_TG

ローカル・ノードのトポロジー・データベース内のエントリーの使用に関する統計情報 (ネットワーク・ノード上) を取得するには、以下の verb を使用します。

- 照会 NN_TOPOLOGY_STATS

リモート LU への通信パスの検査

特定のターゲット LU にアクセスできることを確認するには (LU を所有するノードがアクティブであり、LU への通信パスがあること) を確認するには、次の verb を使用します。

- エイシング

Managing servers and clients on the CS Linux LAN

To obtain a list of servers (nodes) on the CS Linux LAN, use the following verb:

- QUERY_NODE_ALL

To obtain detailed information about a particular node, use the following verb:

- QUERY_NODE

To find out which servers are acting as the controlling configuration file server and backup servers, use the following verb:

- QUERY_SNA_NET

To add new backup servers to the list or to remove existing servers from the list so that they can no longer act as controller servers, use the following verbs:

- ADD_BACKUP, DELETE_BACKUP

To obtain a list of Remote API Clients (on AIX, Linux or Windows) using a particular server on the CS Linux LAN, use the following verb:

- QUERY_RAPI_CLIENTS

Managing configuration file header information

To add a descriptive comment string to the domain configuration file, use the following verb:

- DEFINE_DOMAIN_CONFIG_FILE

To obtain information about the CS Linux version number for which the domain configuration file was created or about the comment string stored in it, use the following verb:

- QUERY_DOMAIN_CONFIG_FILE

There are no corresponding verbs for the node configuration file because the header information in the node configuration file is for CS Linux internal use only; do not attempt to modify it.

Linux リソース使用の管理

CS Linux が内部データ構造に使用できるカーネル・メモリーの量に制限を設定したり、 STREAMS バッファーに使用可能なメモリーの最大量を指定したりするには、以下の verb を使用します。

- SET_KERNEL_MEMORY_LIMIT、SET_BUFFER_可用性

現在の制限と使用量に関する情報を取得するには、以下の verb を使用します。

- 照会カーネル・メモリー制限、QUERY_BUFFER_可用性

NOF indications

A NOF application can use the REGISTER_INDICATION_SINK verb to request information about changes to the CS Linux configuration or to the status of its resources. CS Linux then sends an indication message to the application each time a change occurs.

For more details of the indications that an application can request, see Chapter 4, “NOF Indications,” on page 603.

Except for CONFIG_INDICATION, NOF_STATUS_INDICATION, and SNA_NET_INDICATION, each indication is returned when a resource of the specified type changes its status. For example, if the application registers for DLC indications, CS Linux sends a DLC_INDICATION message to the application each time a DLC becomes active or inactive.

An indication returns summary information about the change that has occurred. If necessary, the application can then issue the appropriate QUERY_* verb to obtain more detailed information.

If the local node is short of resources, it can temporarily suppress indications and not send them to applications. When the resource shortage condition clears, and the local node subsequently generates an indication of a type that it has previously suppressed, it then sets a parameter on the indication to inform the application that one or more previous indications of this type have been lost. The application should then issue QUERY_* verbs for the appropriate resource type to determine the current state of resources.

For more information about registering to receive indications, see “登録受信側のシンク” on page 557. For more information about individual indications, see Chapter 4, “NOF Indications,” on page 603.

構成の指示

アプリケーションは、特定のターゲット (ドメイン構成ファイル、実行中のノード、または非アクティブ・ノード) の構成に対する変更に関する情報を受信するために登録することができます。 これにより、他の NOF アプリケーションまたは管理プログラムが行うことのできる変更を追跡することができます。 これを行うために、アプリケーションは、要求された指示タイプとして CONFIG_INDICATION を指定して、その他の指示のために登録します。

この指示タイプに関連付けられている特定の VCB 構造はありません。 代わりに、構成に対する変更が発生すると、 CS Linux は、変更を行った NOF verb から完了した VCB のコピーを送信することによって、この変更をアプリケーションに対して示します。

構成指示の詳細については、 603 ページの『構成の指示』を参照してください。

SNA network file indications

An application can register to receive information about changes to the SNA network file `sna.net` on the controller server. This enables the application to keep track of changes to this file that can be made by other NOF applications or by the command-line administration program. To do this, the application registers as for other indications, specifying SNA_NET_INDICATION as the requested indication type.

Two VCB structures are associated with this indication type:

- ADD_BACKUP (indicating that a backup server has been added to the end of the file)
- DELETE_BACKUP (indicating that an unused backup server has been removed from the file)

Registering with a type of SNA_NET_INDICATION will return an ADD_BACKUP indication when a backup server is added or a DELETE_BACKUP indication when a server is deleted; the application does not need to register separately for each of these indications. In each case, the format of the indication is a copy of the completed VCB from the NOF verb that made the change.

For more information about SNA network file indications, see “SNA_NET_INDICATION” on page 653.

NOF 状況表示

CS Linux は、アプリケーションがターゲット・ノードまたはファイルにアクセスできなくなった場合 (ターゲット・コンピューター上の CS Linux ソフトウェアが停止されたか、このコンピューターへの通信パスが失われたため) 、 NOF 状況指示を登録済み NOF アプリケーションに送信します。 アプリケーションが制御構成ファイルからの指示を受け取るように登録されている場合は、別のサーバーがコントローラーとして引き継がれる (したがって、ターゲット・ファイルが制御構成ファイルでなくなる) 場合にも、この指示が戻されます。

アプリケーションは、この指示を受信するために明示的に登録する必要はありません。 CS Linux は、該当するターゲット上の任意のタイプの NOF 指示に対して登録されているすべてのアプリケーションにそれを戻します。 この指示は、アプリケーションが REGISTER_INDICATION_SINK verb (またはアプリケーシ

ョンが複数を発行した場合は、最初の REGISTER_INDICATION_SINK verb に対して) に指定されたコールバック・ルーチンに戻されます。

アプリケーションは、ターゲットが失敗したことを示す標識を受け取ると、関連するターゲット・ハンドルを使用する後続のすべての verb は拒否されます。ただし、 DISCONNECT_NODE または CLOSE_FILE (ターゲット・ハンドルを解放するため) は例外です。 さらに、このターゲット・ハンドル上の指示のための登録は失われます。ターゲットが使用可能になったときに未確定の受信を続行するには、アプリケーションはターゲットに再度接続し、必要な指示について再度登録する必要があります。

NOF 状況指示の詳細については、 603 ページの『第 4 章 NOF Indications』を参照してください。

Chapter 2. Writing NOF Applications

This chapter describes the following:

- Client/Server considerations

UNIX

- AIX or Linux considerations
  - NOF API entry points for Linux
  - Compiling and linking the NOF application

WINDOWS

- Windows considerations
  - NOF API entry points for Windows
  - Compiling and linking the NOF application

[redacted]

- Writing portable applications
- Target (node or file) for NOF verbs, and how they interact with the target
- Ordering and dependencies between NOF verbs
- NOF restrictions based on node configuration
- How to request single or multiple data entries with QUERY_* verbs

クライアント/サーバーの考慮事項

クライアント/サーバー・システムでは、サーバー上で実行されているアプリケーション内のすべての NOF verb を使用できます。 Remote API Client で実行するアプリケーションは、以下のように NOF verb を使用できます。

- QUERY_* verb を使用して、構成情報または状況情報を照会することができます。
- verb を使用してセッションを活動化したり、セッション限度を初期化したり、ロギングおよびクライアント/サーバーのトレースを管理したりすることができます。 これらのコマンドを使用するには、 NOF アプリケーションをユーザー ID **根**で実行するか、または **システム** グループ (AIX) または **スナバ** グループ (Linux) のメンバーであるユーザー ID を使用して実行する必要があります。
- 他の verb を使用して構成を変更したり、リソースを開始または停止したりすることはできません。 NOF アプリケーションで構成を変更したり、リソースを開始または停止したりする必要がある場合は、 Linux 用にそれを作成し、それをサーバー上で実行する必要があります。

AIX or Linux considerations

UNIX

This section describes operating system issues that you need to consider when writing NOF applications for use in the AIX or Linux environment.

NOF API entry points for AIX or Linux

An application accesses the NOF API using the following entry point function calls:

nof
Issues a NOF verb synchronously. CS Linux does not return control to the application until verb processing has finished. All NOF verbs except REGISTER_INDICATION_SINK and UNREGISTER_INDICATION_SINK can be issued through this entry point.

An application can use this entry point only if the application can suspend while waiting for CS Linux to completely process a verb.

nof_async
Issues a NOF verb asynchronously. CS Linux returns control to the application immediately, with a returned value indicating whether verb processing is still in progress or has completed. If the returned value indicates that verb processing is still in progress, CS Linux uses an application-supplied callback routine to return the results of the verb processing. In cases when CS Linux is able to completely process the request, the callback routine will not be invoked.

All NOF verbs can be issued through this entry point. The REGISTER_INDICATION_SINK and UNREGISTER_INDICATION_SINK verbs must be issued through this entry point.

An application must use this entry point if either of the following conditions is true:

- The application needs to receive NOF indications.
- The application cannot suspend while waiting for CS Linux to completely process a verb.

nof_async callback routine
When using the asynchronous NOF API entry point, the application must supply a pointer to a callback routine. CS Linux uses this callback routine both for completion of a verb and also for returning NOF data and status indications.

The `nof` and `nof_async` entry points are defined in the NOF header file `nof_c.h`. Parameter types such as `AP_UINT32`, used in these entry points and in the NOF VCBs, are defined in the common header file `values_c.h`, which is included by the NOF header file `nof_c.h`. Both of these files are stored in `/usr/include/sna` (AIX) or `/opt/ibm/sna/include` (Linux).

Synchronous entry point: nof

An application uses the `nof` entry point to issue a NOF verb synchronously. CS Linux does not return control to the application until verb processing has finished.

Function call

```
        void nof (
                 AP_UINT32      target_handle,
                 void *         nofvcb
                 );
```

提供されるパラメーター

アプリケーションは、 `nof` エントリー・ポイントを使用するときに、以下のパラメーターを提供します。

target_handle
ターゲット CS Linux ノードまたはファイルを識別するためにアプリケーションが使用する ID 。 このパラメーターは、以下のいずれかの方法で提供されます。

- 以下の verb では、このパラメーターは指定されていません。これを `0` (ゼロ)に設定してください。 verb が正常に完了すると、 CS Linux は、ターゲット・ハンドルを VCB パラメーターの `1` つとして戻します。 その後、アプリケーションは、後続の verb に対してターゲット・ハンドルを使用します。
  - CONNECT_NODE (稼働中のノードにアクセスする場合、または CS Linux ソフトウェアが開始されているが、ノードがまだ開始されていないサーバー上のノードにアクセスする場合)
  - OPEN_FILE (ドメイン構成ファイルまたは SNA ネットワーク・データ・ファイルにアクセスする場合)
- 以下の verb では、アプリケーションは NULL 値を提供します。

  - QUERY_NODE_ALL (実行中のノードのリストを取得する場合)
  - 照会集中ログ・ロガー
- その他のすべての NOF verb については、アプリケーションは、 CONNECT_NODE verb または OPEN_FILE verb で戻された値を提供します。

nofvcb
発行される verb のパラメーターが入っている Verb 制御ブロック (VCB) へのポインター。 各 verb の VCB 構造体は、 37 ページの『第 3 章 NOF API verbs』 で説明されています。 これらの構造は、 NOF API ヘッダー・ファイルで定義され `/usr/include/sna/nof_c.h` (AIX) または `/opt/ibm/sna/include/nof_c.h` (Linux)。

注 : NOF VCB には、 "予約済み"とマークされた多くのパラメーターが含まれています。これらの一部は CS Linux ソフトウェアによって内部的に使用され、その他はこのバージョンでは使用されず、将来のバージョンで使用される可能性があります。 アプリケーションは、これらの予約済みパラメーターのいずれにもアクセスを試行してはなりません。代わりに、 VCB の内容全体をゼロに設定して、 verb によって使用される他のパラメーターを設定する前に、これらのパラメーターのすべてがゼロになるようにする必要があります。 これにより、 CS Linux は、内部で使用されるパラメーターを誤って解釈しないようにします。また、これらのパラメーターを使用して新しい機能を提供できる将来の CS Linux バージョンについても、ご使用のアプリケーションが引き続き動作することが保証されます。

VCB の内容をゼロに設定するには、 `memset` を使用します。

```
memset(nofvcb、 0、 sizeof(nofvcb));
```

Returned values

The `nof` entry point does not have a return value. When the call returns, the application should examine the return code in the VCB to determine whether the verb completed successfully and to determine parameters it needs for further verbs. In particular, when the CONNECT_NODE or OPEN_FILE verb completes successfully, the VCB contains the *target_handle* that the application should use when issuing subsequent verbs.

同期エントリー・ポイントの使用

ターゲット・ハンドルごとにいつでも、 1 つの同期 verb のみが未解決であることができます。 同じターゲット・ハンドルの別の同期 verb が進行中の場合は、 1 次戻りコード **状態検査の追加** および 2 次戻りコード `AP_SYNC_PENDING` で同期 verb が失敗します。

Asynchronous entry point: nof_async

An application uses `nof_async` to issue a NOF verb asynchronously. The application also supplies a pointer to a callback routine. CS Linux returns control to the application immediately with a returned value that indicates whether verb processing is still in progress or has completed. In most cases, verb processing is still in progress when control returns to the application. In these cases, CS Linux uses the application-supplied callback routine to return the results of the verb processing at a later time. In some cases, verb processing is complete when CS Linux returns control to the application, so CS Linux does not use the application's callback routine.

関数呼び出し

```
AP_UINT16  nof_async(
                     AP_UINT32      target_handle,
                     void *         nofvcb,
                     NOF_CALLBACK   (*comp_proc),
                     AP_CORR        corr
                    );

typedef void (*NOF_CALLBACK) (
                              AP_UINT32      target_handle,
                              void *         nofvcb,
                              AP_CORR        corr
                              AP_UINT32      indic_length
                              );
```

```
typedef union ap_corr  {
                         void *         corr_p;
                         AP_UINT32      corr_l;
                         AP_INT32       corr_i;
                       } AP_CORR;
```

コールバックが NOF_コールバック 構造体のパラメーターについて詳しくは、23 ページの『nof_async エントリー・ポイントで指定されたコールバック・ルーチン』を参照してください。

Supplied parameters

An application supplies the following parameters when it uses the nof_async entry point:

target_handle
This parameter is supplied in one of the following ways:

- For the following verbs, this parameter is not used; set it to 0 (zero). If the verb completes successfully, CS Linux returns the target handle as one of the VCB parameters. The application then uses the target handle for subsequent verbs.
  - CONNECT_NODE (to access a running node, or to access the node on a server where the CS Linux software is started but the node is not yet started)
  - OPEN_FILE (to access the domain configuration fileor the SNA network data file)
- For the following verbs, the application supplies a null value:
  - QUERY_NODE_ALL (to obtain a list of running nodes)
  - QUERY_CENTRAL_LOGGER
- For all other NOF verbs, the application supplies the value that was returned on the CONNECT_NODE or OPEN_FILE verb.

nofvcb
Pointer to a Verb Control Block (VCB) that contains the parameters for the verb being issued. The VCB structure for each verb is described in Chapter 3, “NOF API verbs,” on page 37. These structures are defined in the NOF API header file /usr/include/sna/nof_c.h (AIX) or /opt/ibm/sna/include/nof_c.h (Linux).

Note : The NOF VCBs contain many parameters marked as "reserved"; some of these are used internally by the CS Linux software, and others are not used in this version but may be used in future versions. Your application must not attempt to access any of these reserved parameters; instead, it must set the entire contents of the VCB to zero to ensure that all of these parameters are zero, before it sets other parameters that are used by the verb. This ensures that CS Linux will not misinterpret any of its internally-used parameters, and also that your application will continue to work with future CS Linux versions in which these parameters may be used to provide new functions.

To set the VCB contents to zero, use memset:

```
memset(nofvcb, 0, sizeof(nofvcb));
```

comp_proc
The callback routine that CS Linux will call when the verb completes. For more information about the requirements for a callback routine, see “nof_async エントリー・ポイントで指定されたコールバック・ルーチン” on page 23.

corr
An optional correlator for use by the application. This parameter is defined as a C union so that the application can specify any of three different parameter types: pointer, 32-bit integer, or 16-bit integer.

CS Linux does not use this value, but passes it as a parameter to the callback routine when the verb completes. This value enables the application to correlate the returned information with its other processing.

戻り値

非同期入り口点は、以下のいずれかの値を戻します。

追加完了
verb は既に完了しています。 アプリケーションは、 VCB 内のパラメーターを調べて、 verb が正常に完了したかどうかを判別することができます。 CS Linux は、この verb に対して指定されたコールバック・ルーチンを呼び出しません。

AP_IN_PROGRESS
verb はまだ完了していません。 アプリケーションは、現行 verb の完了に依存していない限り、他の NOF verb を発行するなど、他の処理を続行できます。 しかし、アプリケーションは、この verb に指定された VCB 内のパラメーターの検査や変更を行わないようにする必要があります。

CS Linux は、提供されたコールバック・ルーチンを呼び出して、 verb 処理が完了した時点を示しますその後、アプリケーションは VCB パラメーターを調べることができます。

Using the asynchronous entry point

When using the asynchronous entry point, note the following:

- If an application specifies a null pointer in the *comp_proc* parameter, the verb will complete synchronously (as though the application issued the verb using the synchronous entry point).
- If the call to `nof_async` is made from within an application callback, specifying a null pointer in the *comp_proc* parameter is not permitted. In such cases, CS Linux rejects the verb with a primary return code value of `AP_PARAMETER_CHECK` and a secondary return code value of `AP_SYNC_NOT_ALLOWED`.
- The application must not attempt to use or modify any parameters in the VCB until the callback routine has been called.
- Multiple verbs do not necessarily complete in the order in which they were issued. In particular, if an application issues an asynchronous verb followed by a synchronous verb, the completion of the synchronous verb does not guarantee that the asynchronous verb has already completed.

nof_async エントリー・ポイントで指定されたコールバック・ルーチン

非同期 NOF API エントリー・ポイントを使用する場合、アプリケーションはコールバック ・ルーチンを指すポインターを提供する必要があります。 CS Linux は、このコールバック・ルーチンを verb の完了のために使用し、また NOF 指示を戻す場合にも使用します。 (REGISTER_INDICATION_SINK verb は、コールバック・ルーチンを指定する非同期 verb としても発行されます。コールバックは、指示が受信されるたびに呼び出されます。 その他の NOF verb の場合は、 verb の完了時に指示を受け取ります。) アプリケーションは、 VCB 内の オペコード パラメーターを調べて、コールバック・ルーチンに含まれているイベントを判別する必要があります。

このセクションでは、 CS Linux がコールバック・ルーチンをどのように使用するか、およびコールバック・ルーチンが実行する必要のある関数を説明

コールバック関数

```
NOF_CALLBACK (*comp_proc);
typedef void (*NOF_CALLBACK) (
                               AP_UINT32      target_handle,
                               void *         nofvcb,
                               AP_CORR        corr
                               AP_UINT32      indic_length
                              );
typedef union ap_corr  {
                         void *         corr_p;
                         AP_UINT32      corr_l;
                         AP_INT32       corr_i;
                       } AP_CORR;
```

提供されるパラメーター

CS Linux は以下のパラメーターを使用してコールバック・ルーチンを呼び出します。

target_handle
NOF 指示の場合、 CS Linux は、 REGISTER_INDICATION_SINK verb で提供されたターゲット・ハンドルを渡します。 verb の完了の場合、このパラメーターは定義されていません。

nofvcb
以下のいずれか。

- NOF 指示の場合は、 CS Linux によって提供される VCB を指すポインター。
- verb が完了するために、アプリケーションによって提供された VCB を指すポインター。 VCB には、 CS Linux によって設定された戻りパラメーターが組み込まれました。

コアー
アプリケーションによって提供される相関関係子の値。 この値を使用すると、アプリケーションは、戻された情報を他の処理と関連付けることができます。

コールバック・ルーチンは、これらのパラメーターのすべてを使用する必要はありません (24 ページの『指示に対するコールバック・ルーチンの使用』で説明されている場合 コールバック・ルーチンは、戻されたパラメーターに対して必要なすべての処理を実行できます。あるいは、単に、 verb が完了したことを NOF アプリケーションに通知する変数を設定するだけです。

戻り値

コールバック関数は、値を返しません。

指示に対するコールバック・ルーチンの使用

アプリケーションは NOF verb 用の VCB を割り振りますが、 CS Linux は指示のために VCB を割り振ります。 そのため、アプリケーションは、コールバック・ルーチン内からのみ VCB 情報にアクセスできます。 VCB ポインターは、コールバック・ルーチンの外部では、 CS Linux がコールバック・ルーチンに提供するものではありません。 アプリケーションは、コールバック・ルーチン内から必要なすべての処理を完了するか、またはこのルーチンの外側で使用する必要がある VCB データのコピーを作成する必要があります。

ターゲット・ハンドルの有効範囲

NOF を使用する必要がある各アプリケーションは、独自のハンドルを取得するために CONNECT_NODE verb を発行する必要があります。 2 つの NOF アプリケーションは、同じ NOF ターゲット・ハンドルを使用できません。

特に、子プロセスを作成するために CONNECT_NODE を発行したアプリケーションが子プロセスを作成する場合、子プロセスは親プロセスによって取得されたターゲット・ハンドルを使用する NOF verb を発行することはできません。 ただし、子プロセスは、別の CONNECT_NODE を発行して、それ自体のターゲット・ハンドルを取得できます。

NOF アプリケーションのコンパイルとリンク

AIX applications

To compile and link 32-bit applications, use the following options:

```
-bimport:/usr/lib/sna/nof_r.exp -I /usr/include/sna
```

To compile and link 64-bit applications, use the following options:

```
-bimport:/usr/lib/sna/nof_r64_5.exp -I /usr/include/sna
```

Linux applications

Before compiling and linking a NOF application, specify the directory where shared libraries are stored, so that the application can find them at run time. To do this, set the environment variable `LD_RUN_PATH` to `/opt/ibm/sna/lib`, or to `/opt/ibm/sna/lib64` if you are compiling a 64-bit application.

To compile and link 32-bit applications, use the following options:

```
-I /opt/ibm/sna/include -L /opt/ibm/sna/lib -lnof -lsna_r -lpthread -lpLiS
```

To compile and link 64-bit applications, use the following options:

```
-I /opt/ibm/sna/include -L /opt/ibm/sna/lib64 -lnof -lsna_r -lpthread -lpLiS
```

The option `-lpLiS` is required only if you will be running the application on a CS Linux server; you do not need to use it if you are building the application on an IBM Remote API Client and it will run only on the client. As an alternative to using this option, you can set the the environment variable `LD_PRELOAD` to `/usr/lib/libpLiS.so` before compiling and linking the application.

Windows について

このセクションでは、 Windows クライアントで使用する NOF アプリケーションを作成するときに考慮する必要があるオペレーティング・システムの問題について説明します

Windows 上の Remote API Client で実行されているアプリケーションは、 NOF QUERY_* verb を使用して構成または状況情報を照会することができますが、他の verb を使用して構成を変更したり、リソースを開始または停止したりすることはできません。 NOF アプリケーションで構成を変更したり、リソースを開始または停止したりする必要がある場合は、 Linux 用にそれを作成し、それをサーバー上で実行する必要があります。

NOF API entry points for Windows

A Windows NOF application accesses the NOF API using the following entry point function calls:

`nof`
Issues a NOF verb synchronously. The Remote API does not return control to the application until verb processing has finished.

An application can use this entry point only if the application can suspend while waiting for the Remote API to completely process a verb.

`nof_async`
Issues a NOF verb asynchronously. The Remote API returns control to the application immediately, with a returned value indicating whether verb processing is still in progress or has completed. If the returned value indicates that verb processing is still in progress, it will later complete asynchronously; the Remote API indicates the completion by signaling an event handle supplied by the application. In cases when the Remote API is able to completely process the request, the event handle will not be signaled.

An application must use this entry point if it cannot suspend while waiting for the Remote API to completely process a verb.

The `nof` and `nof_async` entry points are defined in the NOF header file `winnof.h`; this file is installed in the subdirectory in the subdirectory `\sdk` for 32-bit applications, or `\sdk64` for 64-bit applications, within the directory where you installed the Windows Client software. Parameter types such as `AP_UINT32`, used in these entry points and in the NOF VCBs, are defined in the common header file `values_c.h`, which is installed in the same directory and is included by the NOF header file `winnof.h`.

Synchronous entry point: nof

An application uses the `nof` entry point to issue a NOF verb synchronously. The Remote API does not return control to the application until verb processing has finished.

Function call

```
void WINAPI nof (
                 AP_UINT32      target_handle,
                 void *         nofvcb
                );
```

提供されるパラメーター

アプリケーションは、 nof エントリー・ポイントを使用するときに、以下のパラメーターを提供します。

target_handle
ターゲット CS Linux ノードまたはファイルを識別するためにアプリケーションが使用する ID 。 このパラメーターは、以下のいずれかの方法で提供されます。

- 以下の verb では、このパラメーターは指定されていません。これを 0 (ゼロ)に設定してください。 verb が正常に完了すると、リモート API は、 VCB パラメーターの 1 つとしてターゲット・ハンドルを戻します。 その後、アプリケーションは、後続の verb に対してターゲット・ハンドルを使用します。
  - CONNECT_NODE (稼働中のノードにアクセスする場合、または CS Linux ソフトウェアが開始されているが、ノードがまだ開始されていないサーバー上のノードにアクセスする場合)
  - OPEN_FILE (ドメイン構成ファイルまたは SNA ネットワーク・データ・ファイルにアクセスする場合)
- 以下の verb では、アプリケーションは NULL 値を提供します。
  - QUERY_NODE_ALL (実行中のノードのリストを取得する場合)
  - 照会集中ログ・ロガー
- その他のすべての NOF verb については、アプリケーションは、 CONNECT_NODE verb または OPEN_FILE verb で戻された値を提供します。

nofvcb
発行される verb のパラメーターが入っている Verb 制御ブロック (VCB) へのポインター。 各 verb の VCB 構造体は、 37 ページの『第 3 章 NOF API verbs』で説明されています。 これらの構造は、 NOF API ヘッダー・ファイルで定義され ノード・ キュー・ ファイル.

注 : NOF VCB には、 "予約済み"とマークされた多くのパラメーターが含まれています。これらの一部は CS Linux ソフトウェアによって内部的に使用され、その他はこのバージョンでは使用されず、将来のバージョンで使用される可能性があります。 アプリケーションは、これらの予約済みパラメーターのいずれにもアクセスを試行してはなりません。代わりに、 VCB の内容全体をゼロに設定して、 verb によって使用される他のパラメーターを設定する前に、これらのパラメーターのすべてがゼロになるようにする必要があります。 これにより、 CS Linux は、内部で使用されるパラメーターを誤って解釈しないようにします。また、これらのパラメーターを使用して新しい機能を提供できる将来の CS Linux バージョンについても、ご使用のアプリケーションが引き続き動作することが保証されます。

VCB の内容をゼロに設定するには、 memset を使用します。

```
memset(nofvcb、 0、 sizeof(nofvcb));
```

Returned values

The nof entry point does not have a return value. When the call returns, the application should examine the return code in the VCB to determine whether the verb completed successfully and to determine parameters it needs for further verbs. In particular, when the CONNECT_NODE or OPEN_FILE verb completes successfully, the VCB contains the *target_handle* that the application should use when issuing subsequent verbs.

Using the synchronous entry point

Only one synchronous verb can be outstanding at any time for each target handle. A synchronous verb fails with the primary return code AP_STATE_CHECK and secondary return code AP_SYNC_PENDING if another synchronous verb for the same target handle is in progress.

Asynchronous entry point: nof_async

An application uses nof_async to issue a NOF verb asynchronously. The application also supplies a pointer to a callback routine. The Remote API returns control to the application immediately with a returned value that indicates whether verb processing is still in progress or has completed. In most cases, verb processing is still in progress when control returns to the application. In these cases, the Remote API uses the application-supplied callback routine to return the results of the verb processing at a later time. In some cases, verb processing is complete when the Remote API returns control to the application, so the Remote API does not use the application's callback routine.

Function call

```
AP_UINT16 WINAPI nof_async(
                           AP_UINT32      target_handle,
                           void *         nofvcb,
                           NOF_CALLBACK   (*comp_proc),
                           AP_CORR        corr
                          );

typedef void (*NOF_CALLBACK) (
                              AP_UINT32      target_handle,
                              void *         nofvcb,
                              AP_CORR        corr
                              AP_UINT32      indic_length
                              );

typedef union ap_corr  {
                         void *         corr_p;
                         AP_UINT32      corr_l;
                         AP_INT32       corr_i;
                       } AP_CORR;
```

For more information about the parameters in the NOF_CALLBACK structure, see “nof_async エントリー・ポイントで指定されたコールバック・ルーチン” on page 28.

Supplied parameters

An application supplies the following parameters when it uses the nof_async entry point:

target_handle
This parameter is supplied in one of the following ways:

- For the following verbs, this parameter is not used; set it to 0 (zero). If the verb completes successfully, the Remote API returns the target handle as one of the VCB parameters. The application then uses the target handle for subsequent verbs.
  - CONNECT_NODE (to access a running node, or to access the node on a server where the CS Linux software is started but the node is not yet started)
  - OPEN_FILE (to access the domain configuration file or the SNA network data file)
- For the following verbs, the application supplies a null value:
  - QUERY_NODE_ALL (to obtain a list of running nodes)
  - QUERY_CENTRAL_LOGGER
- For all other NOF verbs, the application supplies the value that was returned on the CONNECT_NODE or OPEN_FILE verb.

nofvcb
Pointer to a Verb Control Block (VCB) that contains the parameters for the verb being issued. The VCB structure for each verb is described in Chapter 3, “NOF API verbs,” on page 37. These structures are defined in the NOF API header file nof_c.h.

Note : The NOF VCBs contain many parameters marked as "reserved"; some of these are used internally by the CS Linux software, and others are not used in this version but may be used in future versions. Your application must not attempt to access any of these reserved parameters; instead, it must set the entire contents of the VCB to zero to ensure that all of these parameters are zero, before

it sets other parameters that are used by the verb. This ensures that CS Linux will not misinterpret any of its internally-used parameters, and also that your application will continue to work with future CS Linux versions in which these parameters may be used to provide new functions.

To set the VCB contents to zero, use `memset`:

```
memset(nofvcb, 0, sizeof(nofvcb));
```

comp_proc
The callback routine that the Remote API will call when the verb completes. For more information about the requirements for a callback routine, see “nof_async エントリー・ポイントで指定されたコールバック・ルーチン” on page 28.

corr
An optional correlator for use by the application. This parameter is defined as a C union so that the application can specify any of three different parameter types: pointer, 32-bit integer, or 16-bit integer.

The Remote API does not use this value, but passes it as a parameter to the callback routine when the verb completes. This value enables the application to correlate the returned information with its other processing.

Returned values

The asynchronous entry point returns one of the following values:

AP_COMPLETED
The verb has already completed. The application can examine the parameters in the VCB to determine whether the verb completed successfully. The Remote API does not call the supplied callback routine for this verb.

AP_IN_PROGRESS
The verb has not yet completed. The application can continue with other processing, including issuing other NOF verbs, provided that they do not depend on the completion of the current verb. However, the application should not attempt to examine or modify the parameters in the VCB supplied to this verb.

The Remote API calls the supplied callback routine to indicate when the verb processing completes. The application can then examine the VCB parameters.

Using the asynchronous entry point

When using the asynchronous entry point, note the following:

- If an application specifies a null pointer in the *comp_proc* parameter, the verb will complete synchronously (as though the application issued the verb using the synchronous entry point).
- If the call to `nof_async` is made from within an application callback, specifying a null pointer in the *comp_proc* parameter is not permitted. In such cases, the Remote API rejects the verb with a primary return code value of `AP_PARAMETER_CHECK` and a secondary return code value of `AP_SYNC_NOT_ALLOWED`.
- The application must not attempt to use or modify any parameters in the VCB until the callback routine has been called.
- Multiple verbs do not necessarily complete in the order in which they were issued. In particular, if an application issues an asynchronous verb followed by a synchronous verb, the completion of the synchronous verb does not guarantee that the asynchronous verb has already completed.

nof_async エントリー・ポイントで指定されたコールバック・ルーチン

非同期 NOF API エントリー・ポイントを使用する場合、アプリケーションはコールバック・ルーチンを指すポインターを提供する必要があります。 リモート API は、このコールバック・ルーチンを使用して verb の完了を示します。 このセクションでは、リモート API がコールバック・ルーチンを使用する方法、およびコールバック・ルーチンが実行する必要がある関数について説明し

コールバック関数

```
NOF_CALLBACK (*comp_proc);
typedef void (*NOF_CALLBACK) (
                               AP_UINT32      target_handle,
                               void *         nofvcb,
                               AP_CORR        corr
                               AP_UINT32      indic_length
                              );
typedef union ap_corr  {
                         void *         corr_p;
                         AP_UINT32      corr_l;
                         AP_INT32       corr_i;
                       } AP_CORR;
```

Supplied parameters

The Remote API calls the callback routine with the following parameters:

target_handle
This parameter is undefined.

nofvcb
A a pointer to the VCB supplied by the application. The VCB now includes the returned parameters set by the Remote API.

corr
The correlator value supplied by the application. This value enables the application to correlate the returned information with its other processing.

The callback routine need not use all of these parameters. It can perform all the necessary processing on the returned parameters, or it can simply set a variable to inform the NOF application that the verb has completed.

戻り値

コールバック関数は、値を返しません。

Scope of target handle

Each application that needs to use NOF must issue the CONNECT_NODE verb to obtain its own handle. No two NOF applications can use the same NOF target handle.

NOF アプリケーションのコンパイルとリンク

このセクションでは、 Windows での NOF アプリケーションのコンパイルとリンクについて 説明します。

構造パッキングのコンパイラー・オプション

NOF verb の VCB 構造体はパックされていません。 このパッキング方式を変更するコンパイラー・オプションを使用しないでください。

ドワード パラメーターは DWORD 境界上にあり、 ワード パラメーターは WORD 境界上にあり、 バイト パラメーターはバイト (BYTE) 境界にあります。

ヘッダー・ファイル

Windows NOF アプリケーションに含まれる NOF ヘッダー・ファイルには、 **ノード・キュー・ファイル**という名前が付けられます。 このファイルは、 Windows ソフトウェア上に Remote API Client をインストールしたディレクトリー内の、 32 ビット・アプリケーション用のサブディレクトリー \sdk (64 ビット・アプリケーション用 \sdk64 にインストールされます。

ロード・タイム・リンク

ロード時にアプリケーションを NOF にリンクするには、その TP を API ライブラリー・ファイルにリンクします \ sdk\winnof32.lib for 32-bit applications, or \sdk64\winnof32.lib for 64-bit applications.

実行時リンク

実行時にアプリケーションを NOF にリンクするには、 TP 内に以下の呼び出しを組み込みます。

- **ロード・ライブラリー** は、 NOF 動的リンク・ライブラリーをロードします winnof32.dll
- ProcAddress **の取得** は、必要な各 NOF エントリー・ポイント (nof および/または **非同期**) を指定します。
- ライブラリーが不要になった場合は **フリーライブラリー**

Writing portable applications

The following guidelines are provided for writing CS Linux NOF applications so that they will be portable to other environments:

- Include the NOF header file without any path name prefix. This enables the application to be used in an environment with a different file system. Use include options on the compiler to locate the file (see “NOF アプリケーションのコンパイルとリンク ” on page 24 or “NOF アプリケーションのコンパイルとリンク” on page 29).
- Use the symbolic constant names for parameter values and return codes, not the numeric values shown in the header file; this ensures that the correct value will be used regardless of the way these values are stored in memory.
- Include a check for return codes other than those applicable to your current operating system (for example using a "default" case in a switch statement), and provide appropriate diagnostics.
- Ensure that any parameters shown as reserved are set to 0 (zero).

Target for NOF verbs

A NOF verb can be directed to any of the following targets:

- A running node (to manage the node's resources)
- The node on a server where the CS Linux software is running but where the node has not been started (to start the node, to query the node's stored configuration, or to modify the configuration so that the changes take effect when the node is restarted)
- The domain configuration file (to manage domain resources)
- The sna.net file (to manage the CS Linux servers that can act as backup controllers if the controller server is not available)

The target for a particular NOF verb is identified by the *target_handle* parameter used on the NOF call. An application acquires a target handle using different NOF verbs depending on the target, as follows:

Running node or node on running server
The application issues CONNECT_NODE, specifying the name of the required node, with a null target handle; CS Linux returns a target handle for this node as one of the VCB parameters for CONNECT_NODE.

Domain configuration file
The application issues OPEN_FILE with a null target handle; CS Linux returns a target handle for the file as one of the VCB parameters for OPEN_FILE.

sna.net file
The application issues OPEN_FILE with a null target handle; CS Linux returns a target handle for the file as one of the VCB parameters for OPEN_FILE.

Some NOF verbs can be issued only to particular target types:

- DEFINE_NODE cannot be issued to a running node; it must be issued to a server where the node is not running.
- Verbs associated with node resources, such as DEFINE_LOCAL_LU, must be issued to a node.
- START_* and STOP_* verbs, to start and stop node resources, must be issued to a running node.
- Verbs associated with domain resources must be issued to the domain configuration file.
- Different QUERY_* verbs return information about the definition of a resource, on its current status, or on both definition and status. Status information can only be obtained from a running node. Verbs that return only status information cannot be issued to an inactive node, and verbs that return both definition and status will return only definition information when issued to an inactive node. For example, QUERY_PARTNER_LU_DEFINITION can be issued either to an inactive node (to determine the stored configuration) or to a running node (to determine the current definition). However, QUERY_PARTNER_LU (which returns information about the LU's current sessions) can be issued only to a running node. QUERY_LS (which returns both the definition of the LS and its current status) can be issued either to an inactive node or to a running node, but status information is not returned if you issue it to an inactive node. The description of each QUERY_* verb in Chapter 3, “NOF API verbs,” on page 37 includes information about the valid target types for the verb.
- Verbs associated with managing backup servers (ADD_BACKUP, DELETE_BACKUP, QUERY_SNA_NET, and REGISTER_INDICATION_SINK or UNREGISTER_INDICATION_SINK for SNA network file indications) must be issued to the `sna.net` file.

処理モード

アプリケーションによって使用される各ターゲット・ハンドルには、 NOF verb SET_PROCESSING_MODE を使用して変更できる関連処理モードがあります。 このモードは、アプリケーションのファイル・ロックおよびアクセス許可を制御します。

クライアント上で実行されている NOF アプリケーションの場合、使用可能なモードは読み取り専用モードのみです。 このモードでは、 QUERY_* verb のみが使用可能になります。 リソースの構成または状況を変更するその他のすべての verb は拒否されます。 これにより、アプリケーションはリソースの構成または状況を確認することはできますが、変更することはできません。

サーバー上で実行されている NOF アプリケーションの場合、以下のモードを使用できます。

アプムモード読み取り専用
このモードでは、 QUERY_* verb のみが使用可能になります。 リソースの構成または状況を変更するその他のすべての verb は拒否されます。

これは、ターゲット・ハンドルが最初に割り当てられるときのデフォルト・モードです。これにより、アプリケーションは、リソースの構成または状況を検査することはできますが、変更することはできません。

追加の READ_WRITE
このモードでは、リソースの構成または状況を変更するすべての NOF verb が使用可能になります。

追加モード・コミット
このモードは、ターゲット・ハンドルがドメイン構成ファイル (ノードへの verb の発行時ではなく) を識別する場合にのみ使用できます。 このファイルは、このアプリケーションのみがアクセスできるように、ファイル上のロックを取得します。このファイル・ロックにより、このアプリケーションによって発行された一連の verb の間に他のプロセスによってファイルが変更されないようにすることができます。 また、ファイル・ロックは、 verb の完全なシーケンスが発行されるまで (アプリケーションが **追加モード・コミット** モードから他のモードのいずれか 1 つに変更されるまで) 、ファイルへの変更が行われないことも保証します。

このモードでは、他のプログラムがファイルにアクセスするのを防ぐため、このモードは必要な長さだけ使用する必要があります。 アプリケーションは、ファイルを変更するために必要なすべての verb を直ちに発行してから、他のいずれかのモードに変更する必要があります。

ファイル・ロックを取得できない場合 (例えば、別のプログラムが現在ファイルを変更しているため) 、 SET_PROCESSING_MODE verb は失敗します。

注 : ファイルへの読み取り/書き込みまたはコミット・アクセスを取得するには、ユーザーの NOF アプリケーションが、 SNA 管理者グループ **スナバ** のメンバーであるユーザー ID (または **根**として実行されている) を使用して実行されている必要があります。 ユーザー ID がこのグループまたは **根**のメンバーではない場合、有効な処理モードは **アプムモード読み取り専用**のみです。

NOF verb 間の順序付けと依存関係

NOF verb の順序に関する主な制約事項は、特定のリソースへの最初の参照が、そのリソースの DEFINE_* verb になければならないということです。 これは、以下の依存関係につながります

- 新規ノード構成ファイルの作成時に発行される最初の verb は DEFINE_NODE でなければなりません。
- DLC は、その DLC を参照するポートの前に定義する必要があります。
- ポートは、そのポートを参照する LS または CN の前に定義する必要があります。
- COS を参照するモードの前に、 COS を定義する必要があります。
- PU 名は、この PU を参照する従属 LU の前に (LS 定義の一部として) 定義されていなければなりません。
- LU は、それを含む LU プールの前に定義する必要があります。
- ダウンストリーム PU 名 (LS 定義の一部として) とホスト LU は、それらを参照するダウンストリーム LU の前に定義する必要があります。
- リソースは、 START_* verb が参照する前に定義されている必要があり、 STOP_* verb がそれを参照する前に開始する必要があります。

さらに、実行中のノードを変更するときには、 DEFINE_* verb を使用して (前の定義を変更するために) 2 回目の verb を使用することはできません。 これらの verb の一部については、 2 番目の定義が有効でない (リソースを削除してから再度定義する必要があります)、他の定義では、 2 番目の定義は、リソースが現在非アクティブである場合にのみ有効です。 37 ページの『第 3 章 NOF API verbs』 の個々の DEFINE_* verb の説明は、 2 番目の定義が有効であるかどうかについての情報を提供します。 ドメイン構成ファイルを変更する場合、前の定義を変更するために、必ず 2 番目の DEFINE_* verb を使用することができます。

新規ノード構成ファイルの作成時に発行される最初の verb は DEFINE_NODE でなければなりません。 これには、ノードに関連付けられているすべてのリソースについて、 DEFINE_* verb および SET_ * verb が続いている必要があります。

ドメイン構成ファイルには、ドメイン・リソース・レコードの順序付けに関する制限はありません。

ノード構成に基づく NOF 制約事項

DEFINE_NODE verb には、ノードによってサポートされる機能の範囲を定義するパラメーターが含まれています。 いくつかの NOF verb は、ノードがサポートすることもサポートできないオプションの機能に関連しています。これらの verb は、関連する機能をサポートするノードに対して発行された場合にのみ有効です。

このセクションでは、どの NOF verb を使用できるかに影響を与えるオプション機能を要約しています。これらの機能について詳しくは、 144 ページの『DEFINE_NODE』 を参照してください。

APPN end node and LEN node restrictions

The CS Linux local node can be an APPN network node, an APPN branch network node, an APPN end node, or a LEN node.

The following NOF verbs are only valid at a network node, branch network node, or end node; the primary return code AP_FUNCTION_NOT_SUPPORTED is returned if you attempt to issue them at a LEN node.

- DEFINE_CN
- DELETE_CN
- QUERY_CN
- QUERY_CN_PORT

The following NOF verbs are only valid at a network node or branch network node; the primary return code AP_FUNCTION_NOT_SUPPORTED is returned if you attempt to issue them at an end node or LEN node.

- QUERY_ADJACENT_NN
- QUERY_ISR_SESSION
- QUERY_NN_TOPOLOGY_NODE
- QUERY_NN_TOPOLOGY_STATS
- QUERY_NN_TOPOLOGY_TG
- REGISTER_INDICATION_SINK for any of the following indications:
  - ISR_INDICATION
  - NN_TOPOLOGY_NODE_INDICATION
  - NN_TOPOLOGY_TG_INDICATION

Multiple Domain Support (MDS) restrictions

The local node can be run with or without Multiple Domain Support (MDS). The following NOF verbs are only valid at a node running with MDS; the primary return code AP_FUNCTION_NOT_SUPPORTED is returned if you attempt to issue them at a node without MDS.

- QUERY_ACTIVE_TRANSACTION
- QUERY_MDS_APPLICATION
- QUERY_MDS_STATISTICS

SNA gateway and DLUR restrictions

The local node can be run with or without support for SNA gateway or DLUR or both.

The following NOF verbs are valid only if the node is running with SNA gateway enabled; the primary return code AP_FUNCTION_NOT_SUPPORTED is returned if you attempt to issue them at a node without SNA gateway.

- DEFINE_DOWNSTREAM_LU, DEFINE_DOWNSTREAM_LU_RANGE
- DELETE_DOWNSTREAM_LU, DELETE_DOWNSTREAM_LU_RANGE

The following NOF verbs are valid only if the node is running with DLUR enabled; the primary return code AP_FUNCTION_NOT_SUPPORTED is returned if you attempt to issue them at a node without DLUR.

- DEFINE_DLUR_DEFAULTS
- DEFINE_INTERNAL_PU, DELETE_INTERNAL_PU
- START_INTERNAL_PU, STOP_INTERNAL_PU
- QUERY_DLUR_LU, QUERY_DLUR_PU, QUERY_DLUS

The following NOF verbs are valid only if the node is running with SNA gateway or DLUR or both enabled; the primary return code AP_FUNCTION_NOT_SUPPORTED is returned if you attempt to issue them at a node without either of these two functions.

- QUERY_DOWNSTREAM_LU, QUERY_DOWNSTREAM_PU

List options for QUERY_* Verbs

A NOF application can obtain information about a particular CS Linux resource by issuing a QUERY_* verb for the appropriate resource type. For example, it can obtain information about the configuration of an LS by issuing QUERY_LS. These verbs can either return information about a specific resource (for example, the configuration of a particular LS) or about many resources of the same type (for example, a summary of all configured LSs), depending on the options used. In addition, some QUERY_* verbs have the option of returning either summary or detailed information about the specified resources. This section explains how to use these options.

1 つのリソースまたは複数のリソースに関する情報の入手

QUERY_* verb によって戻される情報は、リソースの名前に従って順序付けされたリスト形式で保管されているものと考えることができます。 例えば、 QUERY_LS によって戻される情報は、 LS 名の順序になっています。 リストの通常の順序は、以下のとおりです。

- 名前の長さによる (最短の名前)
- 同じ長さの名前に対する ASCII 辞書順の順序付け

リストの順序がこの点と異なる場合 (例えば、リストが数値によって配列されている場合) 、この違いは、 37 ページの『第 3 章 NOF API verbs』の個々の動詞の説明で示されます。

これは、完全なリストまたは指定された部分を要求することによって、アプリケーションが複数のリソースに関する情報を取得できることを意味します。 以下の QUERY_* verb のパラメーターは、リストから戻される項目を判別します。

buf_size
戻された情報を受信するためにアプリケーションが提供するデータ・バッファーのサイズ。

num_entries
情報が戻されるリソースの最大数。 アプリケーションは、 1 を指定して、範囲ではなく特定のエントリーを要求するか、範囲を要求するために 1 より大きい数を指定するか、または 0 (ゼロ) を指定して、できるだけ多くのエントリーを要求することができます。

list_options
必要な最初の項目のリスト内の位置は、次のとおりです。

- リストの最初の項目
- 特定の名前付きエントリーから始まる項目
- 特定の名前付き項目の後の次の項目から開始される項目。 (The name specified gives the starting position according to the list ordering and need not exist in the list; for example, if the list contains entries ノード, ノード, ノーデッド, ノード F, and the application requests entries starting from the first entry after ノーデック, the first entry returned is ノーデッド.)

さらに、 *list_options* パラメーターが最初の項目からの開始を要求していない場合は、リスト内の特定の項目の名前が、必要な項目の開始位置を示すために使用されます。

戻される項目の数は、以下の値のうちの最小値です。

- *num_entries* パラメーター(これがゼロ以外の場合)
- 指定されたデータ・バッファーが保持できるエントリーの最大数
- 指定された開始位置からリスト終了までの間の項目数

さらに、この verb は、使用可能な項目の総数と、すべての項目を一度に戻すために必要なバッファーのサイズについての情報を戻します。 アプリケーションが、必要なすべての情報をまだ受け取っていない場合は、それ以降の verb を発行して残りの情報を入手することができます。

これらのオプションにより、アプリケーションは受信する情報を以下のように管理できます。

- To obtain a specific entry, it sets the index value to the name of that entry, *list_options* to indicate "指定された項目から開始", *buf_size* to at least the size of a single entry, and *num_entries* to 1.

- To obtain a complete list a few entries at a time, it first sets *list_options* to indicate "リストの先頭から開始", and uses either *buf_size* or *num_entries* to limit the amount of information returned. If the returned values indicate that there is more information available, it then issues another verb with *list_options* indicating "次の項目から開始する" and sets the index value to the name of the last entry received; this second verb then returns the next section of the list. アプリケーションは、必要な項目をすべて受信するまで、このプロセスを繰り返します。

要約情報または詳細情報の入手

一部の QUERY_* verb は、指定されたリソースに関する要約情報または詳細情報のいずれかを戻すオプションを提供します。 例えば、 QUERY_LOCAL_LU は、 LU 名と LU 別名 (要約情報) だけを戻すか、または LU アドレスおよびセッション限度 (詳細情報) などの追加情報を戻すことができます。 37 ページの『第 3 章 NOF API verbs』 の各 QUERY_* verb の説明は、 verb に要約情報または詳細情報を返すオプションが含まれているかどうかを示します。

このオプションを提供する verb については、 *list_options* パラメーターを使用して、要約情報または詳細情報が必要かどうか、およびリスト内の開始位置を指定します。 これらのオプションを指定するには、論理 **それとも** 演算を使用して 2 つの値を結合します (1 つの値で、リスト内の開始位置を指定し、 1 つの値を指定して、要約情報または詳細情報が必要かどうかを指定します) 、 *list_options* パラメーターをこれら 2 つの値の組み合わせに このオプションを提供しない動詞については、単に *list_options* を単一値に設定して、リスト内の開始位置を示すことができます。

Chapter 3. NOF API verbs

This chapter provides the following information for each NOF API verb:

- Description of the verb's purpose and usage
- Whether the verb can be issued to an active node, an inactive node, the domain configuration file, or the SNA network data file (unless otherwise stated, verbs may be issued either to an active node or to an inactive node)
- Verb control block (VCB) structure, as defined in the NOF API header file `nof_c.h`
- Parameters supplied to the verb by the application
- Parameters returned to the application
- Error return codes for unsuccessful execution

Most parameters supplied to and returned by the NOF interface are hexadecimal values. To simplify coding, these values are represented by meaningful symbolic constants defined in the header file `values_c.h`, which is included by the NOF header file `nof_c.h`. For example, the *opcode* parameter of the ACTIVATE_SESSION verb is the hexadecimal value represented by the symbolic constant `AP_ACTIVATE_SESSION`. The file `values_c.h` also includes definitions of parameter types such as `AP_UINT16` that are used in the NOF VCBs.

It is important that you use the symbolic constant and not the hexadecimal value when setting values for supplied parameters, or when testing values of returned parameters. This is because different Linux systems store these values differently in memory, so the value shown may not be in the format recognized by your system.

The error return codes described in this chapter are specific to each verb. Additional return codes, which are common to all NOF API verbs, are described in Appendix B, “共通戻りコード,” on page 665.

NOF API indications, which the application can accept by registering using the REGISTER_INDICATION_SINK verb, are described separately in Chapter 4, “NOF Indications,” on page 603.

Note : The NOF VCBs contain many parameters marked as "reserved"; some of these are used internally by the CS Linux software, and others are not used in this version but may be used in future versions. Your application must not attempt to access any of these reserved parameters; instead, it must set the entire contents of the VCB to zero to ensure that all of these parameters are zero, before it sets other parameters that are used by the verb. This ensures that CS Linux will not misinterpret any of its internally-used parameters, and also that your application will continue to work with future CS Linux versions in which these parameters may be used to provide new functions.

To set the VCB contents to zero, use `memset`:

```
memset(nofvcb, 0, sizeof(nofvcb));
```

セッションのアクティブ化

ACTIVATE_SESSION verb は、指定されたモードを使用して、ローカル LU と指定されたパートナー LU との間のセッションを活動化するように CS Linux を要求します。 *cnos*_許容値 が **類人猿**に設定されていない場合は、 ACTIVATE_SESSION verb を発行する前に INITIALIZE_SESSION_LIMIT verb を発行する必要があります。

この verb は実行中のノードに対して発行する必要があります

この verb は、クライアント上で実行される NOF アプリケーションから発行できます。 AIX または Linux クライアント上で実行される場合、 NOF アプリケーションは、ユーザー ID **根**で実行するか、または **システム** グループ (AIX) または **スナバ** グループ (Linux) のメンバーであるユーザー ID を使用して実行する必要があります。

VCB 構造体

```
typedef struct activate_session
{
      AP_UINT16        opcode;          /* verb operation code             */
      unsigned char    reserv2;         /* reserved                        */
      unsigned char    format;          /* reserved                        */
      AP_UINT16        primary_rc;      /* primary return code             */
      AP_UINT32        secondary_rc;    /* secondary return code           */
      unsigned char    lu_name[8];      /* local LU name                   */
      unsigned char    lu_alias[8];     /* local LU alias                  */
      unsigned char    plu_alias[8];    /* partner LU alias                */
      unsigned char    mode_name[8];    /* mode name                       */
      unsigned char    fqplu_name[17];  /* fully qualified partner LU name */
      unsigned char    polarity;        /* requested session polarity      */
      unsigned char    session_id[8];   /* session ID                      */
      unsigned char    cnos_permitted;  /* is implicit CNOS permitted?     */
      unsigned char    reserv4[15];     /* reserved                        */
} ACTIVATE_SESSION;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
アクティブ・アクティブ・セッション

lu_name
CS Linux に定義されている、ローカル LU の LU 名。 これは 8 バイトのタイプ A の EBCDIC ストリングで、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。 LU が LU 名の代わりに LU 別名で定義されていることを示すには、このパラメーターを 8 進ゼロに設定します。

lu_alias
CS Linux に定義されている、ローカル LU の LU 別名。 これは 8 バイトの ASCII ストリングで、ローカルで表示可能な文字を使用し、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。 このパラメーターは、 *lu_name* がゼロに設定されている場合のみ使用されます

LU 名と LU 別名の両方がすべてゼロに設定されている場合、 verb は CP に関連付けられた LU (デフォルトの LU) に転送されます。

plu_alias
パートナー LU の LU 別名。 これは 8 バイトの ASCII ストリングで、ローカルで表示可能な文字を使用し、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。 パートナー LU が LU 別名の代わりに完全修飾 LU 名によって定義されていることを示すには、このパラメーターを 8 桁の 2 進ゼロに設定します。

モード名
LU によって使用されるモードの名前。 これは 8 バイトの英数字のタイプ A の EBCDIC ストリング (文字で始まる) で、名前が 8 バイトより短い場合は、右側に EBCDIC のスペースが埋め込まれます。

fqplu_name
CS Linux に対して定義されている、パートナー LU の完全修飾 LU 名。 このパラメーターは、 *plu_alias* フィールドがゼロに設定されている場合のみ使用されます。 *plu_alias* が指定されている場合は無視されます。

この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

極性
セッションの極性。 可能な値は次のとおりです

- 追加ポリシー （ いずれか ）
- AP_POL_FIRST_スピーカー
- 追加ポーリング・ 入札者

追加ポリシー （ **いずれか** ）が設定されている場合、 ACTIVATE_SESSION は最初のスピーカー・セッションを活動化し、それ以外の場合はビッダー・セッションが活動化され AP_POL_FIRST_**スピーカー** または **追加ポーリング・ 入札者** が設定されている場合、 ACTIVATE_SESSION は要求された極性のセッションが使用可能な場合にのみ成功します。

***cnos_*許容値**
CNOS 処理が許可されることを示します。 可能な値は次のとおりです

類人猿
CNOS 処理が許可されます。

アプ・ ノー
CNOS 処理は許可されません。

指定されたモードのセッション限度がリセットされ、このパラメーターが **類人猿**に設定されているために、新規セッションの活動化ができない場合は、暗黙の CNOS 処理によってセッション限度が初期化されます。 CNOS 処理がアクティブのときに、このコマンドの実行は中断されます。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

secondary_rc
可能な値は次のとおりです

折衝済みのアプ
セッションは正常に活動化されました。このモードに定義されたセッション限度は、活動化プロセス中に折衝されました。

指定されたアプ
セッションは正常に活動化されました。セッション限度は変更されませんでした。

セッション *ID*
アクティブ化されたセッションの 8 バイトの ID 。

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

許可される最大値 （ _MAX__ ）
セッションを活動化できません。これは、この LU - LU モードの組み合わせの現行のセッション限度を超えるためです。

エイプ無効ルリエイリアス
lu_alias パラメーターが、定義されたローカル LU 別名と一致しませんでした。

ファイル名の変更
lu_name パラメーターが、定義されたローカル LU 名と一致しませんでした。

ファイル名の変更
fqplu_name パラメーターが、定義されたどのパートナー LU 名とも一致しなかったか、 *plu_alias* パラメーターが定義されたパートナー LU 名と一致しませんでした。

許可されている AP_INVALID_CNOS_缶
*cnos_*許容値 パラメーターに指定された値が無効でした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: 活動化障害

他のエラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターのいずれかを戻します。

primary_rc
可能な値は次のとおりです

追加アクティブ・エラーの再試行
アクションを必要とする条件 (構成の不一致やセッション・プロトコル・エラーなど) が原因で、セッションを活動化できませんでした。 CS Linux ログ・ファイルでエラー条件についての情報を確認し、この verb を再試行する前に訂正してください。

追加活動失敗再試行
一時的条件 (リンク障害など) が原因で、セッションを活動化できませんでした。 この verb を再試行してください。できれば、タイムアウト後に条件がクリアされるようにしてください。 CS Linux ログ・ファイルで、エラー状態に関する情報を確認してください。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

バックアップの追加

アプリケーションは、この verb を使用して、 **スネネット** ファイル内のバックアップ・サーバーのリストにサーバーを追加します。これにより、現行コントローラーが非アクティブになった場合に、このサーバーが制御構成ファイル・サーバーとして機能することができます。 新しいサーバーがリストの末尾に追加されるため、ファイルにリストされている他のすべてのサーバーが非アクティブの場合にのみ、このサーバーはコントローラーになります。

This verb must be issued to the **スネネット** ファイル。

VCB 構造体

```
typedef struct add_backup
{
      AP_UINT16        opcode;             /* verb operation code            */
      unsigned char    reserv2;            /* reserved                       */
      unsigned char    format;             /* reserved                       */
      AP_UINT16        primary_rc;         /* primary return code            */
      AP_UINT32        secondary_rc;       /* secondary return code          */
      unsigned char    backup_name[128];   /* name of backup server to add   */
      unsigned char    reserv3[4];         /* reserved                       */
} ADD_BACKUP;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_ADD_BACKUP

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

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

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

secondary_rc
未使用。

戻りパラメーター: 状態チェック

状態チェックのために verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc
可能な値は次のとおりです

レコードの追加 （ _R ）
指定されたサーバー名は、既にファイルにリストされています。

ファイルの検証ターゲット
NOF API 呼び出しのターゲット・ハンドルに、構成ファイルまたはノードが指定されています。 This verb must be issued to the **スネネット** ファイル。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **状態検査の追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

追加 DLC_TRACE

この verb は、 DLC で送信される SNA メッセージのトレースを指定します。 これを使用して、特定の DLC、ポート、 LS、または HPR RTP 接続、または指定された LS 上の特定のセッションでトレースを活動化し、どのタイプのメッセージをトレースするかを指定することができます。 また、すべての DLC、ポート、リンク・ステーション、および HPR RTP 接続でトレースを活動化するために使用することもできます。 CS Linux トレースの使用方法について詳しくは、「 *IBM Communications Server for Linux* 管理ガイド上のデータ・センター・デプロイメント」を参照してください。

同じリソースに関連する複数の ADD_DLC_TRACE verb が発行された場合、メッセージは、現在アクティブないずれかの verb と一致するとトレースされます。 例えば、

- ポートとその LS に関するすべてのメッセージをトレースするために verb を発行した場合に、 2 番目の verb を発行して、ポートが所有する LSs のいずれかに指定された LFSID が指定されたメッセージのみをトレースする場合、 LS のすべてのメッセージはトレースされ続けます (最初の verb と一致するため)。 その後、 REMOVE_DLC_TRACE を使用してポートのトレースを除去すると、指定された LFSID を持つ LS 上のメッセージは引き続きトレースされます (これらのメッセージは依然としてアクティブの 2 番目の verb と一致するため)。ただし、この LS 上の他のメッセージはトレースされません。
- すべてのリソースで XID メッセージをトレースするために動詞を発行した後、特定の LS 上の SC および DFC メッセージをトレースするために 2 番目の verb を発行すると、この LS の 3 つのメッセージ・タイプがトレースされます。

SDLC 回線をトレースしていて、より詳細なトレース情報を必要とする場合は、回線トレースだけでなく、 SDLC の内部トレースを使用してこれを入手することができます。 追加の詳細情報は、回線トレースの出力の一部としてフォーマットされるため、 1 つのファイルにすべての SDLC トレースが表示されます。 詳しくは、 582 ページの『SET_TRACE_TYPE』を参照してください。

注 : SET_TRACE_TYPE verb には、トレース・ファイル内の各項目を指定された長さまで切り捨てるためのオプションが含まれています。 このオプションは、 DLC トレースと、 SET_TRACE_TYPE によって指定されたカーネル・コンポーネント・トレースに適用されます。

VCB structure

```
typedef struct add_dlc_trace
{
      AP_UINT16           opcode;             /* verb operation code         */
      unsigned char       reserv2;            /* reserved                    */
      unsigned char       format;             /* reserved                    */
      AP_UINT16           primary_rc;         /* primary return code         */
      AP_UINT32           secondary_rc;       /* secondary return code       */
      DLC_TRACE_FILTER    filter;             /* resource to be traced       */
} ADD_DLC_TRACE;

typedef struct dlc_trace_filter
{
      unsigned char       resource_type;      /* type of resource            */
      unsigned char       resource_name[8];   /* name of resource            */
      SNA_LFSID           lfsid;              /* session identifier          */
      unsigned char       message_type;       /* type of messages            */
} DLC_TRACE_FILTER;

typedef struct sna_lfsid
{
      union
      {
         AP_UINT16        session_id;
         struct
         {
           unsigned char  sidh;
           unsigned char  sidl;
         } s;
      } uu;
      AP_UINT16           odai;
} SNA_LFSID;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
追加 DLC_TRACE

ファイル . リソース・タイプ
トレースするリソースを指定します。また、オプションで、このリソースについてトレースされる 特定のメッセージ・タイプを指定します。 可能な値は次のとおりです

リソースの追加 （ _R ）
すべての DLC、ポート、リンク・ステーション、および HPR RTP 接続のトレース・オプションをセットアップします。

アプドゥルク
リソース名で指定されている DLC のトレース・オプションをセットアップし、この DLC を使用するすべてのポートおよび LS についてトレース・オプションをセットアップします。

ポート （ ポート ）
リソース名で指定されたポート、およびこのポートを使用するすべての LSs について、トレース・オプションをセットアップします。

類人猿
リソース名で指定された LS のトレース・オプションをセットアップします。

ファイル・ ソース・ タイプの追加
リソース名で指定された RTP 接続のトレース・オプションを指定します。

PORT_DEFINED_LS の追加
リソース名で指定されたポートのトレース・オプションをセットアップし、このポートを使用するすべての定義済み LSs (ただし、暗黙の LSs ではない) を設定します。

PORT_IMPLICIT_LS
リソース名で指定されたポートのトレース・オプションをセットアップし、このポートを使用するすべての暗黙的な LSs (ただし定義されていない LSs) のトレース・オプションをセットアップします。

ファイル . リソース名
トレースが活動化されている DLC、ポート、 LS、または RTP 接続の名前。 リソース・タイプ が **リソースの追加** （ _R ）に設定されている場合、このパラメーターは予約済みです

リソース・タイプ が **ファイル· ソース· タイプの追加**に設定されている場合は、特定の RTP 接続の名前 (この名前は @ 文字で始まる) を指定するか、またはこのパラメーターをすべてゼロに設定して、すべての RTP トラフィックがトレースされることを示すことができます。

filter.lfsid
指定された LS 上のセッションのローカル・フォーム・セッション ID 。 これは リソース・タイプ **類人猿**の場合にのみ有効です。この場合、このセッションのメッセージのみをトレースすることを示します。 この構造体には、 QUERY_SESSION verb の SESSION_STATS セクションに戻される以下の 3 つの値が含まれています。

filter.lfsid.uu.s.sidh
セッション ID の高位バイト。

filter.lfsid.uu.s.sidl
セッション ID の低位バイト。

filter.lfsid.o ダイ
出荷元宛先の割り当てインディケーター。

filter.message_type
指定されたリソースまたはセッションについてトレースするメッセージのタイプ 。 すべてのメッセージをトレースするには、このパラメーターを **すべての** TRACE_ALL に設定するか、以下の 1 つ以上の値 (論理 **それとも**を使用して結合) を指定します。

追加 TRACE_XID
XID メッセージ

追加の TRACE_SC
セッション制御 RU

追跡中のデータの追加
データ・フロー制御 RU

追加トレース· コマンド
FMD メッセージ

トレース· セット （ _S ）
RH が含まれていない非 BBIU セグメント

追加の TRACE_CTL
MU および XID のその他のメッセージ

追加 TRACE_NLP
ネットワーク層プロトコル・メッセージのトレース

追加の TRACE_NC
ネットワーク制御メッセージのトレース

For tracing on an RTP connection, the values **追加** TRACE_XID, **追加** TRACE_NLP, and **追加の** TRACE_CTL are ignored. RTP トレースには、リストされた他の値の少なくとも 1 つを指定する必要があります。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

secondary_rc
Not used.

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_RESOURCE_TYPE
The *resource_type* parameter specified a value that was not valid.

AP_INVALID_MESSAGE_TYPE
The *message_type* parameter specified a value that was not valid.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

APING

APING is the APPN version of the "ping" utility; it allows a management application to check the communications path from a local LU to a remote LU in the network.

CS 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, the APING verb returns information about the time taken to allocate a conversation to the partner LU and to send and receive data.

The application must supply a VCB that is large enough to include a partner TP verification string of the requested size as well as the basic APING VCB structure; the returned data includes this string appended to the end of the basic structure.

This verb 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 verb must be issued to a running node.

VCB structure

```
typedef struct aping
{
     AP_UINT16         opcode;           /* verb operation code            */
     unsigned char     reserv2;          /* reserved                       */
     unsigned char     format;           /* reserved                       */
     AP_UINT16         primary_rc;       /* primary return code            */
     AP_UINT32         secondary_rc;     /* secondary return code          */
     unsigned char     lu_name[8];       /* local LU name                  */
     unsigned char     lu_alias[8];      /* local LU alias                 */
     AP_UINT32         sense_data;       /* sense data                     */
     unsigned char     plu_alias[8];     /* partner LU alias               */
     unsigned char     mode_name[8];     /* mode name                      */
     unsigned char     tp_name[64];      /* destination TP name            */
     unsigned char     security;         /* security level                 */
     unsigned char     reserv3a[3];      /* reserved                       */
     unsigned char     pwd[10];          /* password                       */
     unsigned char     user_id[10];      /* user ID                        */
     AP_UINT16         dlen;             /* length of data to send         */
     AP_UINT16         consec;           /* number of consecutive sends    */
     unsigned char     fqplu_name[17];   /* fully qualified partner LU name */
```

```
    unsigned char     echo;              /* data echo flag                 */
    AP_UINT16         iterations;        /* number of iterations           */
    AP_UINT32         alloc_time;        /* time taken for ALLOCATE        */
    AP_UINT32         min_time;          /* minimum send/receive time      */
    AP_UINT32         avg_time;          /* average send/receive time      */
    AP_UINT32         max_time;          /* maximum send/receive time      */
    AP_UINT16         partner_ver_len;   /* size of string to receive      */
} APING;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_APING

lu_name
LU name of the local LU. This is an 8-byte type-A EBCDIC string, padded on the right with spaces if the name is shorter than 8 characters. To indicate that the LU is identified by its LU alias instead of its LU name, set this parameter to 8 binary zeros and specify the LU alias in the following parameter.

lu_alias
LU alias of the local LU. This parameter is used only if the *lu_name* field is set to 8 binary zeros, and is ignored otherwise. The alias is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes. To use the default LU (the LU associated with the CP), set both the *lu_name* and *lu_alias* parameters to 8 binary zeros.

plu_alias
Partner LU alias. This should be the alias of an LU on a remote node; you are not recommended to use APING with a partner LU on the local node.

The alias is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes. To indicate that the LU is identified by its fully qualified name instead of its alias, set this parameter to 8 binary zeros and specify the LU name in the *fqplu_name* parameter.

mode_name
Name of the mode used by the LU pair. This is an 8-byte type-A EBCDIC string (starting with a letter), padded on the right with spaces if the name is shorter than 8 characters.

tp_name
Name of the invoked TP (generally set to APINGD). This is a 64-byte string, padded on the right with spaces.

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

AP_NONE
No security information is required.

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

AP_PGM
A user ID and password are required to start the TP.

AP_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 AP_PGM. This is a 10-byte type-AE EBCDIC character string, padded on the right with EBCDIC spaces if the password is shorter than 10 bytes.

user_id
User ID required to access the partner TP; this parameter is required only if the security parameter is set to AP_SAME or AP_PGM. This is a 10-byte type-AE EBCDIC character string, padded on the right with EBCDIC spaces if the user ID is shorter than 10 bytes.

dlen
Length of the data string to be sent to the partner LU. (The NOF API application does not need to provide a data string; the APING TP simply sends a string of zeros of the specified length.)

consec
Number of consecutive data strings sent to the partner LU during each iteration. The APING TP sends this number of data strings, each containing the number of bytes specified by the *dlen* parameter. It then requests either data or a confirmation message from the partner TP, depending on the setting of the *echo* parameter.

fqplu_name
Fully qualified network name for the partner LU. This parameter is used only if the *plu_alias* field is set to 8 binary zeros, and is ignored otherwise. This should be the name of an LU on a remote node; you are not recommended to use APING with a partner LU on the local node.

The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

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

AP_YES
After sending the specified number of data strings, APING waits to receive data from the partner LU.

AP_NO
After sending the specified number of data strings, APING requests confirmation from the partner LU, but does not receive data.

iterations
Number of times that the APING TP should perform the sequence of sending data to the partner LU and requesting either data or confirmation.

partner_ver_len
Maximum length of the partner TP verification data string which can be received by the NOF API application. The application must supply a VCB large enough to include this string as well as the basic APING VCB structure, because the string will be appended to the returned VCB.

Returned parameters: successful execution

If the verb executes successfully, APING returns the following parameters:

primary_rc
AP_OK

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 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 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 zero, this parameter is not used.

partner_ver_len
Length of verification string returned by the partner TP.

In addition to these returned parameters, the verification string returned by the partner TP is appended to the end of the APING VCB. The length of this string is given by *partner_ver_len*. If *partner_ver_len* is zero, then this string is not returned.

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

エイプ無効ルリエイリアス
lu_alias パラメーターが、定義済みの LU 別名と一致しませんでした。

ファイル名の変更
lu_name パラメーターが、定義済みの LU 名と一致しませんでした。

アプリケーション・ バッドセキュリティー
担保 パラメーターが有効な値に設定されていませんでした。

認識されていないパートナー・ モード
plu_alias、 *fqplu_name*、または モード名 に指定された値が、定義済みのパートナー LU またはモードと一致しませんでした。

アプリケーション・ パートナーの LU__ エイリアス
plu_alias に指定された値が、定義済みのパートナー LU と一致しませんでした

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: allocation failure

If the verb does not execute because CS Linux cannot allocate the APPC conversation, CS Linux returns the following parameters:

primary_rc
`AP_ALLOCATION_ERROR`

secondary_rc
Possible values are:

AP_ALLOCATION_FAILURE_NO_RETRY
The conversation cannot 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 verb until the error has been corrected.

AP_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 verb, preferably after a timeout to allow the condition to clear.

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

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

AP_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 verb until the cause of the error has been corrected.

AP_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 verb, preferably after a timeout to allow the condition to clear.

sense_data
If the *secondary_rc* parameter is AP_ALLOCATION_FAILURE_NO_RETRY, this parameter contains the SNA sense data associated with the error. For all other *secondary_rc* values, this parameter is reserved.

戻りパラメーター: 会話障害

パートナー TP との APPC 会話が失敗したために verb が実行されなかった場合、 CS Linux は以下のパラメーターを戻します。

primary_rc

非 CONV_FAILURE_NO_RETRY
セッション・プロトコル・エラーなどの永続的な状態が原因で、会話が終了しました。 エラー・ログ・ファイルを確認して、エラーの原因を判別してください。 エラーが訂正されるまで、APING verb を再試行しないでください。

primary_rc

追加の失敗の再試行
この会話は、一時エラーのために終了しました。 APING verb を再試行してください。 再度問題が発生した場合は、エラー・ログ・ファイルを調べて、エラーの原因を判別してください。

primary_rc

割り振り解除異常終了
パートナー TP は、エラー状態のため、会話を割り振り解除しました。 エラーの理由は、リモート・ノードに記録されている可能性があります。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

CHANGE_SESSION_LIMIT

The CHANGE_SESSION_LIMIT verb requests CS Linux to change the session limits for a particular LU-LU-mode combination. Sessions may be activated or deactivated as a result of processing this verb.

This verb must be issued to a running node.

VCB structure

```
typedef struct change_session_limit
{
   AP_UINT16       opcode;                 /* verb operation code          */
   unsigned char   reserv2;                /* reserved                     */
   unsigned char   format;                 /* reserved                     */
   AP_UINT16       primary_rc;             /* primary return code          */
   AP_UINT32       secondary_rc;           /* secondary return code        */
   unsigned char   lu_name[8];             /* local LU name                */
   unsigned char   lu_alias[8];            /* local LU alias               */
   unsigned char   plu_alias[8];           /* partner LU alias             */
   unsigned char   fqplu_name[17];         /* fully qualified partner      */
                                           /*  LU name                     */
```

```
    unsigned char   reserv3;                /* reserved                     */
    unsigned char   mode_name[8];           /* mode name                    */
    unsigned char   reserv3a;               /* reserved                     */
    unsigned char   set_negotiable;         /* set max negotiable limit?    */
    AP_UINT16       plu_mode_session_limit; /* session limit                */
    AP_UINT16       min_conwinners_source;  /* minimum source contention    */
                                            /*  winner sessions             */
    AP_UINT16       min_conwinners_target;  /* minimum target contention    */
                                            /*  winner sessions             */
    AP_UINT16       auto_act;               /* auto activation limit        */
    unsigned char   responsible;            /* who is responsible for       */
                                            /*  deactivating                */
    unsigned char   reserv4[3];             /* reserved                     */
    AP_UINT32       sense_data;             /* sense data                   */
} CHANGE_SESSION_LIMIT;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_CHANGE_SESSION_LIMIT

lu_name
LU name of the local LU, as defined to CS Linux. This is an 8-byte type-A EBCDIC string, padded on the right with spaces if the name is shorter than 8 bytes. To indicate that the LU is defined by its LU alias instead of its LU name, set this parameter to 8 binary zeros.

lu_alias
LU alias of the local LU, as defined to CS Linux. This is an 8-byte ASCII string, using any locally displayable characters, padded on the right with spaces if the name is shorter than 8 bytes. It is used only if *lu_name* is set to zeros.

To indicate the LU associated with the CP (the default LU), set both *lu_name* and *lu_alias* to 8 binary zeros.

plu_alias
LU alias of the partner LU.

This is an 8-byte ASCII string, using any locally displayable characters, padded on the right with spaces if the name is shorter than 8 bytes. To indicate that the partner LU is defined by its fully qualified LU name instead of its LU alias, set this parameter to 8 binary zeros.

fqplu_name
Fully qualified LU name for the partner LU, as defined to CS Linux. This parameter is used only if the *plu_alias* field is set to zeros; it is ignored if *plu_alias* is specified.

The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

mode_name
Name of the mode to be used by the LUs.

This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces if the name is shorter than 8 bytes.

set_negotiable
Specifies whether the maximum negotiable session limit for this mode should be modified. Possible values are:

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

AP_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 permitted between these two LUs using this mode. 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 verb). This value may be negotiated with the partner LU.

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

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

auto_act
Number of sessions to automatically activate after the session limit is changed. 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 verb). 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 AP_DEACT_NORMAL) below this limit, new sessions are activated up to this limit.

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

AP_SOURCE
The local LU is responsible.

AP_TARGET
The partner LU is responsible.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

secondary_rc
可能な値は次のとおりです

折衝済みのアプ
セッション限度は変更されましたが、 1 つ以上の値がパートナー LU によって折衝されました。

指定されたアプ
セッション限度は、パートナー LU によって折衝されることなく、要求に応じて変更されました。

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

許可される最大値 （ _MAX__ ）
plu_mode_session_limit、 *min_conwinners_source* ソース、 *min_conwinners_target*、または 自動実行 パラメーターが、有効範囲外の値に設定されました。

ゼロに変更されて値を変更
この verb を使用して *plu_mode_session_limit* パラメーターをゼロに設定することはできません。代わりに RESET_SESSION_LIMIT を使用してください。

エイプ無効ルリエイリアス
lu_alias パラメーターが、定義されたローカル LU 別名と一致しませんでした。

ファイル名の変更
lu_name パラメーターが、定義されたローカル LU 名と一致しませんでした。

ファイル・パス名
モード名 パラメーターが定義済みのどのモード名とも一致しませんでした

ファイル名の変更
fqplu_name パラメーターが、定義済みのパートナー LU 名と一致しませんでした。

無責任のアプインバリデータ
責任者 パラメーターが有効な値に設定されていませんでした。

付加的なセットのネゴシエーションが可能
セット折衝可能 パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: state check

If the verb does not execute because of a state error, CS Linux returns the following parameters.

primary_rc
AP_STATE_CHECK

secondary_rc

AP_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.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_STATE_CHECK, which are common to all NOF verbs.

Returned parameters: session allocation error

If the verb does not execute because of a session allocation error, CS Linux returns the following parameters:

primary_rc
AP_ALLOCATION_ERROR

secondary_rc

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

sense_data
The SNA sense data associated with the allocation failure.

Returned parameters: CNOS processing errors

If the verb does not execute because of an error, CS Linux returns the following parameters.

primary_rc

AP_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 CS Linux log file for information about the error condition, and correct it before retrying this verb.

primary_rc
AP_CNOS_PARTNER_LU_REJECT

secondary_rc

AP_CNOS_COMMAND_RACE_REJECT
The verb failed because the specified mode was being accessed by another administration program (or internally by the CS Linux software) for session activation or deactivation, or for session limit processing. The application should retry the verb, preferably after a timeout to allow the race condition to be cleared.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

ファイルのクローズ

アプリケーションは、この verb を使用して、ファイルへの NOF verb の発行を終了したときに、ドメイン構成ファイルまたは **スネネット** ファイルに対してそのハンドルを解放します。 アプリケーションがクローズしようとしているファイルは、呼び出しの *target_handle* パラメーターによって識別されます。

アプリケーションは、オープン・ファイル・ハンドルが終了する前に、常に CLOSE_FILE を発行する必要があります。 verb が正常に完了すると、ファイルを識別するターゲット・ハンドルが有効でなくなります。

この verb は、ドメイン構成ファイルまたは **スネネット** ファイル。 に対して発行される必要があります。

VCB structure

```
 typedef struct close_file
{
       AP_UINT16           opcode;           /* verb operation code         */
       unsigned char       reserv2           /* reserved                    */
       unsigned char       format;           /* reserved                    */
       AP_UINT16           primary_rc;       /* primary return code         */
       AP_UINT32           secondary_rc;     /* secondary return code       */
 } CLOSE_FILE;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
ファイルの追加クローズ

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

secondary_rc
未使用。

Returned parameters: state check

If the verb does not execute because of a state check, CS Linux returns the following parameters:

primary_rc
AP_STATE_CHECK

secondary_rc

AP_VERB_IN_PROGRESS
The specified file cannot be released because a previous verb issued for this target handle is still outstanding. All verbs for the target file must be completed before attempting to close the file.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_STATE_CHECK, which are common to all NOF verbs.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

接続ノード

アプリケーションは、 CS Linux ノード (アクティブまたは非アクティブ) との通信を確立するために、この verb を使用します。 この verb は、ノードを識別するターゲット・ハンドルを戻します。このハンドルは、アプリケーションが他の NOF verb を使用して、 verb のターゲットを示すことができます。

VCB 構造体

```
typedef struct connect_node
{
      AP_UINT16         opcode;          /* verb operation code            */
      unsigned char     reserv2;         /* reserved                       */
      unsigned char     format;          /* reserved                       */
      AP_UINT16         primary_rc;      /* primary return code            */
      AP_UINT32         secondary_rc;    /* secondary return code          */
      unsigned char     node_type;       /* which node to connect to       */
      unsigned char     node_name[128];  /* name of Node                   */
      AP_UINT32         target_handle;   /* handle for subsequent verbs    */
      unsigned char     node_status;     /* node status                    */
      unsigned char     reserv3[12];     /* reserved                       */
} CONNECT_NODE;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_CONNECT_NODE

node_type
To connect to a particular node in order to manage the node's configuration, set this parameter to AP_SPECIFIED_NODE.

To connect to the node currently acting as the central logger, set this parameter to AP_CENTRAL_LOGGER. This value is required if the application will be issuing the following verbs:

- SET_CENTRAL_LOGGING, QUERY_CENTRAL_LOGGING
- SET_GLOBAL_LOG_TYPE, QUERY_GLOBAL_LOG_TYPE
- SET_LOG_FILE, QUERY_LOG_FILE (if central logging is in use)

node_name
Name of the CS Linux node to connect to. This parameter is reserved if *node_type* is set to AP_CENTRAL_LOGGER.

If the node name includes a . (period) character, CS Linux assumes that it is a fully-qualified name; otherwise it performs a DNS lookup to determine the node name.

If CS Linux is running with all components on a single computer, you can set this parameter to all binary zeros; there is no need to specify the node name. Otherwise, setting this parameter to all binary zeros indicates the default local node (on the same CS Linux server as the application).

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

secondary_rc
Not used.

target_handle
Returned value for use on subsequent verbs.

node_status
Specifies the status of the node. Possible values are:

AP_NDE_STARTING
The node is in the process of being activated.

AP_NDE_STARTED
The node is active.

AP_NDE_STOPPING
The node is in the process of being deactivated.

AP_NDE_STOPPED
The node is not active.

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc

AP_INVALID_NODE_NAME
The value that was specified for the *node_name* parameter was not valid.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: state check

If the verb does not execute because of a state error, CS Linux returns the following parameters:

primary_rc
AP_STATE_CHECK

secondary_rc
Possible values are:

AP_CONNECTION_NOT_MADE
An error occurred in connecting to the node.

AP_INVALID_VERSION
The application could not connect to the node, because there was a version mismatch between the CS Linux software on the computer where the application is running and the computer where the target node is defined. If you are in the process of upgrading the network, so that different

computers are running different levels of the CS Linux software, nodes running on the back-level software can be managed only by applications running on the back-level software.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_STATE_CHECK, which are common to all NOF verbs.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

DEACTIVATE_CONV_GROUP

The DEACTIVATE_CONV_GROUP verb requests the deactivation of the session corresponding to the specified conversation group. Although this verb is part of the NOF API, it is primarily intended for use by application programmers writing TPs that use the APPC API. The conversation group identifier is returned by the APPC verbs [MC_]ALLOCATE, [MC_]GET_ATTRIBUTES, and RECEIVE_ALLOCATE.

This verb must be issued to a running node.

VCB 構造体

```
typedef struct deactivate_conv_group
{
      AP_UINT16        opcode;          /* verb operation code              */
      unsigned char    reserv2;         /* reserved                         */
      unsigned char    format;          /* reserved                         */
      AP_UINT16        primary_rc;      /* primary return code              */
      AP_UINT32        secondary_rc;    /* secondary return code            */
      unsigned char    lu_name[8];      /* local LU name                    */
      unsigned char    lu_alias[8];     /* local LU alias                   */
      AP_UINT32        conv_group_id;   /* conversation group identifier    */
      unsigned char    type;            /* deactivation type                */
      unsigned char    reserv3[3];      /* reserved                         */
      AP_UINT32        sense_data;      /* deactivation sense data          */
} DEACTIVATE_CONV_GROUP;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DEACTIVATE_CONV_GROUP

lu_name
LU name of the local LU, as defined to CS Linux. This is an 8-byte type-A EBCDIC string, padded on the right with spaces if the name is shorter than 8 bytes. To indicate that the LU is defined by its LU alias instead of its LU name, set this parameter to 8 binary zeros.

lu_alias
LU alias of the local LU, as defined to CS Linux. This is an 8-byte ASCII string, using any locally displayable characters, padded on the right with spaces if the name is shorter than 8 bytes. It is used only if *lu_name* is set to zeros.

To indicate the LU associated with the CP (the default LU), set both *lu_name* and *lu_alias* to 8 binary zeros.

conv_group_id
Conversation group identifier for the session to be deactivated.

type
Type of deactivation. Possible values are:

AP_DEACT_CLEANUP
Deactivate the session immediately, without waiting for sessions to end.

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

sense_data
If type is set to AP_DEACT_CLEANUP, this parameter specifies the sense data to be used when deactivating the session. Otherwise this parameter is not used.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

エイプデ ACT_CG_INVALID_CGID
conv_group_id パラメーターが、有効な会話グループ ID と一致しませんでした。

ファイルのクリーンアップ・ タイプ （ _R ）
タイプ パラメーターが有効な値に設定されていませんでした。

エイプ無効ルリエイリアス
lu_alias パラメーターが、定義済みの LU 別名と一致しませんでした。

ファイル名の変更
lu_name パラメーターが、定義済みの LU 名と一致しませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

解除 LU_0_TO_3 の非活動化

DEACTIVATE_LU_0_TO_3 verb は、 CS Linux に、 3270 エミュレーションまたは LUA (タイプ 0、 1、 2、または 3 の LU) で使用する特定の LU のセッションを非活動化するように要求します。 CS Linux は、PLU-SLU セッションのために TERM_SELF メッセージをホストに送信することにより、セッションを非活動化します。

この verb は実行中のノードに対して発行する必要があります

VCB structure

```
typedef struct deactivate_lu_0_to_3
{
  AP_UINT16      opcode;                 /* verb operation code          */
  unsigned char  reserv2;
  unsigned char  format;
  AP_UINT16      primary_rc;             /* primary return code          */
  AP_UINT32      secondary_rc;           /* secondary return code        */
  unsigned char  lu_name[8];             /* LU Name                      */
} DEACTIVATE_LU_0_TO_3;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
LU_0_TO_3 の非アクティブ化解除

lu_name
CS Linux に定義されている LU の LU 名。 これは 8 バイトのタイプ A の EBCDIC ストリングで、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters. This return code can also indicate that there was no active session for the specified LU (implying that the session has already been deactivated).

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

ファイル名の変更
lu_name パラメーターが、定義済みの LU 名と一致しませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

DEACTIVATE_SESSION

The DEACTIVATE_SESSION verb requests CS Linux to deactivate a particular session, or all sessions on a particular mode.

This verb must be issued to a running node.

VCB 構造体

```
typedef struct deactivate_session
{
      AP_UINT16         opcode;           /* verb operation code           */
      unsigned char     reserv2;          /* reserved                      */
      unsigned char     format;           /* reserved                      */
      AP_UINT16         primary_rc;       /* primary return code           */
      AP_UINT32         secondary_rc;     /* secondary return code         */
      unsigned char     lu_name[8];       /* local LU name                 */
      unsigned char     lu_alias[8];      /* local LU alias                */
      unsigned char     session_id[8];    /* session identifier            */
      unsigned char     plu_alias[8];     /* partner LU alias              */
      unsigned char     mode_name[8];     /* mode name                     */
      unsigned char     type;             /* deactivation type             */
      unsigned char     reserv3[3];       /* reserved                      */
      AP_UINT32         sense_data;       /* deactivation sense data       */
      unsigned char     fqplu_name[17];   /* fully qualified partner       */
                                          /*  LU name                      */
```

```
      unsigned char     reserv4[20];       /* reserved                        */
} DEACTIVATE_SESSION;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
非アクティブ・セッションの追加

lu_name
CS Linux に定義されている、ローカル LU の LU 名。 これは 8 バイトのタイプ A の EBCDIC ストリングで、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。 LU が LU 名の代わりに LU 別名で定義されていることを示すには、このパラメーターを 8 進ゼロに設定します。

lu_alias
CS Linux に定義されている、ローカル LU の LU 別名。 これは 8 バイトの ASCII ストリングで、ローカルで表示可能な文字を使用し、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。 *lu_name* がゼロに設定されている場合にのみ使用されます。

CP (デフォルト LU) に関連した LU を示すためには、 *lu_name* と *lu_alias* の両方を 2 進ゼロに設定します。

セッション *ID*
非アクティブ化するセッションの 8 バイトの ID 。 このフィールドが 8 進ゼロに設定されている場合、 CS Linux は、パートナー LU とモードのすべてのセッションを非活動化します。

plu_alias
パートナー LU の LU 別名。

これは 8 バイトの ASCII ストリングで、ローカルで表示可能な文字を使用し、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。 パートナー LU が LU 別名の代わりに完全修飾 LU 名によって定義されていることを示すには、このパラメーターを 8 桁の 2 進ゼロに設定します。

モード名
LU によって使用されるモードの名前。

これは 8 バイトの英数字のタイプ A の EBCDIC ストリング (文字で始まる) で、名前が 8 バイトより短い場合は、右側に EBCDIC のスペースが埋め込まれます。

タイプ
非アクティブ化のタイプ。 可能な値は次のとおりです

追加 DEACT_CLEANUP
セッションが終了するのを待たずに、セッションを即時に非活動化します。

通常のアプデ ACT_正常
セッションを使用しているすべての会話が終了するまでセッションを非活動化しないでください。

sense_data
タイプが **追加** DEACT_CLEANUP に設定されている場合、このパラメーターは、セッションを非活動化するときに使用されるセンス・データを指定します。 それ以外の場合は、このパラメーターは使用

fqplu_name
CS Linux に対して定義されている、パートナー LU の完全修飾 LU 名。 このパラメーターは、*plu_alias* フィールドがゼロに設定されている場合のみ使用されます。 *plu_alias* が指定されている場合は無視されます。

この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters. This return code can also indicate that the session ID did not match the session ID of an active session (implying that the session has already been deactivated).

primary_rc
AP_OK

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

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

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

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

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

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

定義された値の定義ノード

DEFINE_ADJACENT_LEN_NODE は、隣接 LEN ノードとそれに関連する LU のためにノード・ディレクトリー・データベースに項目を追加するか、以前に定義された LEN ノードに追加の LU エントリーを追加します。

この verb は、 LEN ノードおよび関連する LUs; に対する一連の DEFINE_DIRECTORY_ENTRY verb と同等です。これは、 LEN ノードの構成を単一の verb で定義する高速方式を提供します。 この verb によって作成されたディレクトリー項目を照会するには、 QUERY_DIRECTORY_ENTRY を使用します。

この verb が LEN ノードのサーバーとして動作するネットワーク・ノードに対して発行される場合、 LEN ノードのリソースはネットワーク・ノードのディレクトリー・データベースに追加されます。 これは、ネットワーク・ノードがネットワーク全体からアクセスできるように、これらのリソースのネットワーク検索に応答することを意味します。 verb がエンド・ノードに対して発行される場合、 LEN ノードのリソースはそのエンド・ノードに対してのみアクセス可能です。

VCB structure

```
typedef struct define_adjacent_len_node
{
      AP_UINT16         opcode;             /* verb operation code          */
      unsigned char     reserv2;            /* reserved                     */
      unsigned char     format;             /* reserved                     */
```

```
      AP_UINT16         primary_rc;          /* primary return code          */
      AP_UINT32         secondary_rc;        /* secondary return code        */
      unsigned char     cp_name[17];         /* CP name                      */
      unsigned char     description[32];     /* resource description         */
      unsigned char     reserv1[16];         /* reserved                     */
      unsigned char     num_of_lus;          /* number of LUs                */
      unsigned char     wildcard_lus;        /* wildcard LUs                 */
      unsigned char     reserv3[8];          /* reserved                     */
      unsigned char     lu_names[10][8];     /* LU names                     */
} DEFINE_ADJACENT_LEN_NODE;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DEFINE_ADJACENT_LEN_NODE

cp_name
The fully qualified name of the CP in the adjacent LEN node. This should match the name the LEN node sends on its XIDs (if it supports them), and the adjacent CP name specified on the DEFINE_LS for the link to the LEN node.

The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

description
A null-terminated text string (0-31 characters followed by a null character) describing the adjacent LEN node. This string is for information only; it is stored in the configuration and returned on the QUERY_DIRECTORY_ENTRY verb, but CS Linux does not make any other use of it.

num_of_lus
The number of LUs to be defined, in the range 0-10. To define an adjacent node with more than 10 LUs, use multiple DEFINE_ADJACENT_LEN_NODE verbs for the same CP name.

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

AP_YES
The specified LU names are wildcard entries.

AP_NO
The specified LU names are explicit entries.

lu_names
The names of the LUs being defined on the LEN node. Each name is an 8-byte type-A EBCDIC character string, right-padded with EBCDIC spaces, 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 above).

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 EBCDIC dot) 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 verb 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 LEN node, use multiple DEFINE_ADJACENT_LEN_NODE verbs.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

ファイルの検証 CP_NAME
cp_name パラメーターに、無効な文字が含まれていました。

ファイル名の変更
指定された LU 名の 1 つ以上に、無効な文字が含まれていました。

類型の無効です。
lus の *num_of_lus* パラメーターが有効範囲内にありませんでした。

ファイル・ ワイルドカード名を使用できます
ワイルドカード *(d_lus)* パラメーターが **類人猿**に設定されましたが、指定された LU 名の 1 つ以上がすでに別の親ノードで定義されていました。

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc
可能な値は次のとおりです

ファイルの検証 CP_NAME
指定された CP 名は、既にディレクトリー項目に定義されており、 LEN ノードではありません。

ファイル名の変更
指定された 1 つ以上の LU 名が、別の親ノード上で既に定義されています。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

DEFINE_CN

DEFINE_CN defines a Connection Network (otherwise known as a Virtual Routing Node or VRN). The verb 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.

DEFINE_CN can be used to redefine an existing Connection Network. In particular, new ports can be added to the list of ports which access the connection network by issuing another DEFINE_CN. (Ports can be removed in the same way by issuing the DELETE_CN verb).

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

VCB structure

```
typedef struct define_cn
{
      AP_UINT16          opcode;              /* verb operation code         */
      unsigned char      reserv2;             /* reserved                    */
```

```
      unsigned char      format;              /* reserved                    */
      AP_UINT16          primary_rc;          /* primary return code         */
      AP_UINT32          secondary_rc;        /* secondary return code       */
      unsigned char      fqcn_name[17];       /* name of connection network  */
      CN_DEF_DATA        def_data;            /* CN defined data             */
      unsigned char      port_name[8][8];     /* port names                  */
} DEFINE_CN;
```

```
typedef struct cn_def_data
{
      unsigned char      description[32];     /* resource description        */
      unsigned char      reserve0[16];        /* reserved                    */
      unsigned char      num_ports;           /* number of ports on CN       */
      unsigned char      cn_type;             /* reserved                    */
      unsigned char      ipv6_addr_only;      /* use IPv6 address            */
      unsigned char      reserve1[14];        /* reserved                    */
      TG_DEFINED_CHARS   tg_chars;            /* TG characteristics          */
} CN_DEF_DATA;
```

```
typedef struct tg_defined_chars
{
      unsigned char      effect_cap;          /* effective capacity          */
      unsigned char      reserve1[5];         /* reserved                    */
      unsigned char      connect_cost;        /* connection cost             */
      unsigned char      byte_cost;           /* byte cost                   */
      unsigned char      reserve2;            /* reserved                    */
      unsigned char      security;            /* security                    */
      unsigned char      prop_delay;          /* propagation delay           */
      unsigned char      modem_class;         /* reserved                    */
      unsigned char      user_def_parm_1;     /* user-defined parameter 1    */
      unsigned char      user_def_parm_2;     /* user-defined parameter 2    */
      unsigned char      user_def_parm_3;     /* user-defined parameter 3    */
} TG_DEFINED_CHARS;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
付加されている _CN

fqcn_name
接続ネットワークの完全修飾名。 この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

データの説明の説明
接続ネットワークを記述するヌルで終了するテキスト・ストリング (0 から 31 文字の後にヌル文字を続けたストリング) 。 このストリングは情報専用です。このストリングはノードの構成ファイルに保管され、 QUERY_CN verb で戻されますが、 CS Linux はそれ以外の使用を行いません。

データ・ポートの不良 *.num_port*
この verb に含まれるポートの数。各 DEFINE_CN verb は、最大 8 個のポートを指定できます。 8 個を超えるポートを持つ CN を定義するには、同じ CN 名に対して複数の DEFINE_CN verb を発行します。 CN 上のポートの最大合計数は 239 です。

def_data.ipv6_address_only
HPR/IP の IPv6 ネットワーク上で接続ネットワークを定義する場合、このパラメーターは、接続ネットワークの IP アドレッシングが IPv6 DNS 名のみを使用するか、 IPv6 アドレスのみを使用するかを示します。 可能な値は次のとおりです

そうだ
Connection Network の IP アドレス指定は、 IPv6 アドレスのみを使用します。

違う。
接続ネットワークの IP アドレス指定は、 IPv6 DNS 名のみを使用します。

***データ .tg_chars.effect_cap* を定義しています**
1 秒当たりの実際のビット数 (回線速度)。 この値は、 1 バイトの浮動小数点数としてエンコードされます。この数値は、「 0.1 mmm * 2 イーエー という式で表されます。この場合、バイトのビット表記は b'eeeeemmm'です。 有効容量の各単位は、 300 ビット / 秒に等しくなります。

***def_data.tg_chars.connect_*原価**
接続時間当たりのコスト。 有効な値は、 0-255 の範囲内の整数値です。ここで、 0 は接続時間あたりのコストが最も低く、 255 が最大値です。

***データ . tg_chars.byte_*原価**
バイト単位のコスト。 有効な値は、 0-255 の範囲の整数値です。ここで、 0 はバイト当たりの最小コストで、 255 は最大値です。

def_data.tg_chars.security
ネットワークのセキュリティー・レベル。 可能な値は次のとおりです

SEC_SEC_安全でない
セキュリティーなし。

パブリック・ スイッチ・ スケジュールのネットワークの切り替え
データは公衆交換網を介して伝送される。

セキュリティー・ アンダーグラウンド・ ケーブル （ _H ）
データは安全な地下ケーブルで送信されます。

セキュリティー・ セキュリティーの AP_SEC_SECURE_CONDUIT
保護されていないセキュア・コンジットでは、回線を介してデータが伝送されます。

セキュリティー・ セキュリティー・ コンジットの追加
データは、物理的な盗聴に対して保護されているコンジットの行を介して伝送されます。

セキュリティー暗号化 （ _H ）
データは、回線を介して伝送される前に暗号化

SEC_SEC_GUARDED_放射線
物理および放射線のタップに対して保護されている回線を介してデータが伝送されます。

def_data.tg_chars.prop_delay
伝搬遅延: シグナルがリンクの長さを移動するのにかかる時間。 リンクのタイプに応じて、以下の値のいずれかを指定します。

AP_PROP_DELAY_最小値
最小伝搬遅延。

AP_PROP_DELAY_LAN （ R ）
遅延は 480 マイクロ秒未満 (LAN の場合の標準) より小さくなります。

AP_PROP_DELAY_電話
遅延は 480 から 49,512 マイクロ秒の範囲です (電話ネットワークの場合は標準です)。

AP_PROP_DELAY_PKT_SWITCHED_NET
遅延は、 49,512 から 245,760 マイクロ秒の範囲です (パケット交換ネットワークの場合は標準)。

AP_PROP_DELAY_衛星
遅延は 245,760 マイクロ秒より大きくなります (サテライト・リンクの場合は標準)。

AP_PROP_DELAY_最大値
最大伝搬遅延。

***def_data.tg_chars.user_def_parm_1* から *def_data.tg_chars.user_def_parm_3* まで**
上記のパラメーターでカバーされていない他の TG 特性を組み込むために使用できる、ユーザー定義パラメーター。 これらの各パラメーターは、 0-255 の範囲内の値に設定する必要があります。

ポート名
接続ネットワーク上で定義されている最大 8 つのポート名の配列。 各ポート名は 8 バイトの ASCII ストリングであり、名前が 8 バイトより短い場合は右側にスペースが埋め込まれ、 DEFINE_PORT verb で既に定義されている必要があります。 ポート・タイプは、接続ネットワーク (イーサネット、トークンリング、エンタープライズ・エクステンダー) をサポートするネットワーク・タイプでなければな

りません。 新しいポート名を指定する別の DEFINE_CN を発行することによって、接続ネットワーク上で追加のポートを定義できます。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameter:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

リンク・ リンクの検証 / セキュリティーが無効です
担保 パラメーターが、有効な値のいずれかに設定されていませんでした。

許可される最大値 （ _MAX__ ）
指定されたポート数を追加すると、 CN 上のポートの最大合計数を超えます。

ファイル・ ファイル名の変更
fqcn_name パラメーターに、無効な文字が含まれているか、または正しい形式ではありませんでした。

指定された AP_INVALID_NUM_PORTS_ネス
ポート数 パラメーターが有効な値に設定されていませんでした。

ポートフォリオ・ ポート名
指定された 1 つ以上のポート名が、定義されたポートの名前と一致しませんでした。

ファイル・ ポート・ タイプの追加
指定されたポートの DLC タイプがネットワーク・タイプではなく Point - to - Point タイプ (SDLC など) であるため、指定されたポートの 1 つ以上を CN にすることはできません。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: state check

If the verb does not execute because of a state error, CS Linux returns the following parameters:

primary_rc
AP_STATE_CHECK

secondary_rc
Possible values are:

AP_PORT_ACTIVE
The specified port cannot be modified because it is currently active.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_STATE_CHECK, which are common to all NOF verbs.

戻りパラメーター: 関数はサポートされません

ローカル・ノードが LEN ノードであるために verb が正常に実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc

付加機能がサポートされていません
ローカル・ノードは LEN ノードです。 この verb は、ネットワーク・ノードまたはエンド・ノードでのみ有効です。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

定義さないコア

DEFINE_COS は、サービス・クラス定義を追加するか、以前に定義された COS を変更します。 この定義では、ノードおよび TG 特性の範囲を、経路計算に使用される重みに関連付ける TG "行" およびノード "行"を指定します。 重みが低いほど、経路は良好になります。

VCB structure

The DEFINE_COS verb contains a variable number of `cos_tg_row` and `cos_node_row` structures; the number of each is specified by the *num_of_node_rows* and *num_of_tg_rows* parameters. The TG rows are included at the end of the main DEFINE_COS structure, in ascending order of weight; they are followed by the node rows, again in ascending order of weight.

```
typedef struct define_cos
{
   AP_UINT16          opcode;                 /* verb operation code        */
   unsigned char      reserv2;                /* reserved                   */
   unsigned char      format;                 /* reserved                   */
   AP_UINT16          primary_rc;             /* primary return code        */
   AP_UINT32          secondary_rc;           /* secondary return code      */
   unsigned char      cos_name[8];            /* class of service name      */
   unsigned char      description[32];        /* resource description       */
   unsigned char      reserv1[16];            /* reserved                   */
   unsigned char      transmission_priority;  /* transmission priority      */
   unsigned char      reserv3[9];             /* reserved                   */
   unsigned char      num_of_node_rows;       /* number of node rows        */
   unsigned char      num_of_tg_rows;         /* number of TG rows          */
} DEFINE_COS;
```

```
typedef struct cos_tg_row
{
   TG_DEFINED_CHARS   minimum;                /* minimum                    */
   TG_DEFINED_CHARS   maximum;                /* maximum                    */
   unsigned char      weight;                 /* weight                     */
   unsigned char      reserv1;                /* reserved                   */
} COS_TG_ROW;
```

```
typedef struct tg_defined_chars
{
   unsigned char      effect_cap;             /* effective capacity         */
   unsigned char      reserve1[5];            /* reserved                   */
   unsigned char      connect_cost;           /* cost per connect time      */
   unsigned char      byte_cost;              /* cost per byte              */
   unsigned char      reserve2;               /* reserved                   */
   unsigned char      security;               /* security                   */
   unsigned char      prop_delay;             /* propagation delay          */
   unsigned char      modem_class;            /* reserved                   */
   unsigned char      user_def_parm_1;        /* user defined parameter 1   */
   unsigned char      user_def_parm_2;        /* user defined parameter 2   */
   unsigned char      user_def_parm_3;        /* user defined parameter 3   */
} TG_DEFINED_CHARS;
```

```
typedef struct cos_node_row
{
   COS_NODE_STATUS     minimum;               /* minimum                    */
   COS_NODE_STATUS     maximum;               /* maximum                    */
   unsigned char       weight;                /* weight                     */
```

```
   unsigned char       reserv1;                /* reserved                   */
} COS_NODE_ROW;
```

```
typedef struct cos_node_status
{
   unsigned char       rar;                    /* route additional resistance*/
   unsigned char       status;                 /* node status                */
   unsigned char       reserv1[2];             /* reserved                   */
} COS_NODE_STATUS;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DEFINE_COS

cos_name
Class of service name. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces.

description
A null-terminated text string (0-31 characters followed by a null character) describing the COS. This string is for information only; it is stored in the node's configuration file and returned on the QUERY_COS verb, but CS Linux does not make any other use of it.

transmission_priority
Transmission priority. Possible values are:

AP_LOW

AP_MEDIUM

AP_HIGH

AP_NETWORK

num_of_node_rows
Number of node rows which follow the DEFINE_COS VCB (after the TG rows). The maximum is 8.

num_of_tg_rows
Number of TG rows which follow the DEFINE_COS VCB. The maximum is 8.

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 concatenated in ascending order of weight.

cos_tg_row.minimum.effect_cap
Minimum limit for actual bits per second rate (line speed). The value is encoded as a 1-byte floating point number, represented by the formula 0.1 mmm * 2 ^{eeeee} where the bit representation of the byte is b'eeeeemmm'. Each unit of effective capacity is equal to 300 bits per second.

cos_tg_row.minimum.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.

cos_tg_row.minimum.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.

cos_tg_row.minimum.security
Minimum level of security. Possible values are:

AP_SEC_NONSECURE
No security.

AP_SEC_PUBLIC_SWITCHED_NETWORK
Data is transmitted over a public switched network.

AP_SEC_UNDERGROUND_CABLE
Data is transmitted over secure underground cable.

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

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

AP_SEC_ENCRYPTED
Data is encrypted before transmission over the line.

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

cos_tg_row.minimum.prop_delay
Minimum limits for 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:

AP_PROP_DELAY_MINIMUM
Minimum propagation delay.

AP_PROP_DELAY_LAN
Delay is less than 480 microseconds (typical for a LAN).

AP_PROP_DELAY_TELEPHONE
Delay is in the range 480-49,512 microseconds (typical for a telephone network).

AP_PROP_DELAY_PKT_SWITCHED_NET
Delay is in the range 49,512-245,760 microseconds (typical for a packet-switched network).

AP_PROP_DELAY_SATELLITE
Delay is greater than 245,760 microseconds (typical for a satellite link).

AP_PROP_DELAY_MAXIMUM
Maximum propagation delay.

cos_tg_row.minimum.user_def_parm_1 **through** ***cos_tg_row.user_def_parm_3***
Minimum values for user-defined parameters, which you can use to include other TG characteristics not covered by the above parameters. Each of these parameters must be set to a value in the range 0-255.

cos_tg_row.maximum.effect_cap
Maximum limit for actual bits per second rate (line speed). The value is encoded as a 1-byte floating point number, represented by the formula 0.1 mmm * 2 ^{eeeee} where the bit representation of the byte is b'eeeeemmm'. Each unit of effective capacity is equal to 300 bits per second.

cos_tg_row.maximum.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.

cos_tg_row.maximum.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.

cos_tg_row.maximum.security
Maximum level of security. Possible values are:

AP_SEC_NONSECURE
No security.

AP_SEC_PUBLIC_SWITCHED_NETWORK
Data is transmitted over a public switched network.

AP_SEC_UNDERGROUND_CABLE
Data is transmitted over secure underground cable.

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

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

AP_SEC_ENCRYPTED
Data is encrypted before transmission over the line.

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

cos_tg_row.maximum.prop_delay
Maximum limits for 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:

AP_PROP_DELAY_MINIMUM
Minimum propagation delay.

AP_PROP_DELAY_LAN
Delay is less than 480 microseconds (typical for a LAN).

AP_PROP_DELAY_TELEPHONE
Delay is in the range 480-49,512 microseconds (typical for a telephone network).

AP_PROP_DELAY_PKT_SWITCHED_NET
Delay is in the range 49,512-245,760 microseconds (typical for a packet-switched network).

AP_PROP_DELAY_SATELLITE
Delay is greater than 245,760 microseconds (typical for a satellite link).

AP_PROP_DELAY_MAXIMUM
Maximum propagation delay.

cos_tg_row.maximum.user_def_parm_1 **through** ***cos_tg_row.maximum.user_def_parm_3***
Maximum values for user-defined parameters, which you can use to include other TG characteristics not covered by the above parameters. Each of these parameters must be set to a value in the range 0-255.

cos_tg_row.weight
Weight associated with this TG 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.

cos_node_row.minimum.rar
Route additional resistance minimum. Values must be in the range 0-255.

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

AP_UNCONGESTED
The number of ISR sessions is below the *isr_sessions_upper_threshold* value in the node's configuration.

AP_CONGESTED
The number of ISR sessions exceeds the threshold value.

cos_node_row.maximum.rar
Route additional resistance maximum. Values must be in the range 0-255.

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

AP_UNCONGESTED
The number of ISR sessions is below the *isr_sessions_upper_threshold* value in the node's configuration.

AP_CONGESTED
The number of ISR sessions exceeds the threshold value.

cos_node_row.weight
Weight associated with this node row.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

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

AP_INVALID_NUMBER_OF_NODE_ROWS
The *num_of_node_rows* parameter was not in the valid range.

AP_INVALID_NUMBER_OF_TG_ROWS
The *num_of_tg_rows* parameter was not in the valid range.

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

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc

最大余弦の表 （ フル ）
新しい COS を定義することはできません。これは、 (DEFINE_NODE の *cos_cache_size* の場合 パラメーターで指定される) ノードで許可される COS 定義の最大数を超えるためです。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **状態検査の追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

キュー・サイド情報の定義

この verb は、サイド情報エントリーを追加または置換します。 CPI-C サイド情報エントリーは、会話特性のセットをシンボリック宛先名と関連付けます。 この verb で提供されたものと同じシンボリック宛先名を持つサイド情報エントリーが既に存在する場合、この情報は、この呼び出しに提供されたデータで上書きされます。

この verb と CPI-C 関数 CPICIC_Side_Information の設定との違いに注意してください。 この verb は、ドメイン構成ファイルを変更し、すべての CS Linux CPI-C アプリケーションに影響を与えます。 CPI-C 関数は、サイド情報テーブルのメモリー内のアプリケーション独自のコピーを変更し、その他の CPI-C アプリケーションには影響を与えません。

この verb は、ドメイン構成ファイルに対して発行する必要があります。

VCB 構造体

```
typedef struct define_cpic_side_info
{
   AP_UINT16                opcode;             /* verb operation code       */
   unsigned char            reserv2;            /* reserved                  */
   unsigned char            format;             /* reserved                  */
   AP_UINT16                primary_rc;         /* primary return code       */
   AP_UINT32                secondary_rc;       /* secondary return code     */
   unsigned char            reserv2a[8];        /* reserved                  */
   unsigned char            sym_dest_name[8];   /* Symbolic destination name */
   CPIC_SIDE_INFO_DEF_DATA  def_data;
} DEFINE_CPIC_SIDE_INFO;
```

```
typedef struct cpic_side_info_def_data
{
   unsigned char    description[32];              /* resource description    */
   unsigned char    reserv1[16];                 /* reserved                */
   CPIC_SIDE_INFO   side_info;                   /* CPIC side info          */
   unsigned char    user_data[24];               /* reserved                */

} CPIC_SIDE_INFO_DEF_DATA;
```

```
typedef struct cpic_side_info
{
   unsigned char    partner_lu_name[17];          /* Fully qualified         */
                                                  /*  partner LU name        */
   unsigned char    reserved[3];                 /* Reserved                */
   AP_UINT32        tp_name_type;                /* TP name type            */
   unsigned char    tp_name[64];                 /* TP name                 */
   unsigned char    mode_name[8];                /* Mode name               */
   AP_UINT32        conversation_security_type;  /* Conversation security   */
                                                  /*  type                   */
   unsigned char    security_user_id[10];        /* User ID                 */
   unsigned char    security_password[10];       /* Password                */
   unsigned char    lu_alias[8];                 /* LU alias                */
} CPIC_SIDE_INFO;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DEFINE_CPIC_SIDE_INFO

sym_dest_name
Symbolic destination name which identifies the side information entry. This is an 8-byte ASCII string, padded on the right with spaces if necessary. The name can contain any displayable character.

def_data.description
A null-terminated text string (0-31 characters followed by a null character) describing the side information entry. This string is for information only; it is stored in the configuration file and returned on the QUERY_CPIC_SIDE_INFO verb, but CS Linux does not make any other use of it.

def_data.side_info.partner_lu_name
Fully qualified name of the partner LU. The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

def_data.side_info.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:

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

XC_SNA_SERVICE_TP
Service TP. The TP name must be specified as an 8-character ASCII string representing the hexadecimal digits of a 4-character name. For example, if the hexadecimal representation of the name is 0x21F0F0F8, set the *def_data.side_info.tp_name* parameter to the 8-character string `21F0F0F8'.

The first character (represented by two bytes) must be a hexadecimal value in the range 00-3F, excluding 0E and 0F; the remaining characters (each represented by two bytes) must be valid EBCDIC characters.

def_data.side_info.tp_name
TP name of the target TP. This is a 64-byte ASCII character string, padded on the right with ASCII spaces.

def_data.side_info.mode_name
Name of the mode used to access the target TP. This is an 8-byte ASCII character string, padded on the right with spaces.

def_data.side_info.conversation_security_type
Specifies whether the target TP uses conversation security. Possible values are:

XC_SECURITY_NONE
The target TP does not use conversation security.

XC_SECURITY_PROGRAM
The target TP uses conversation security. The *security_user_id* and *security_password* parameters specified below will be used to access the target TP.

XC_SECURITY_PROGRAM_STRONG
As for XC_SECURITY_PROGRAM, except that the local node must not send the password across the network in clear text format. This value can be used only if the remote system supports password substitution.

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

def_data.side_info.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 XC_SECURITY_NONE.

def_data.side_info.security_password
Password used to access the partner TP. This parameter is required only if the *conversation_security_type* parameter is set to XC_SECURITY_PROGRAM or XC_SECURITY_PROGRAM_STRONG.

def_data.side_info.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: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

定義のデフォルトの T_PU

DEFINE_DEFAULT_PU は、 CS Linux 管理サービス・データを処理するためのデフォルトの PU を指定します。 各ノードごとに 1 つのデフォルト PU のみを定義することができます。異なる PU 名に対する 2 番目の DEFINE_DEFAULT_PU verb は、以前の定義をオーバーライドします。

DEFINE_DEFAULT_PU を使用すると、ユーザーは、デフォルト PU の任意のフィールドを定義、再定義、または変更することができます。 また、この verb では、ヌルの PU 名を指定することによって、デフォルト PU を削除することもできます。

アプリケーションが PU 名を指定せずに MS API verb TRANSFER_MS_DATA を発行した場合、データはローカル・ノードに定義されたデフォルト PU に経路指定され、この PU のセッションでホスト SSCP とのセッションが行われます。 TRANSFER_MS_DATA についての詳細は、 *AIX* または *Linux MS* プログラマーズ・ガイドでの *IBM Communications Server for Data Center* デプロイメントを参照してください。

VCB structure

```
typedef struct define_default_pu
{
      AP_UINT16           opcode;              /* verb operation code       */
      unsigned char       reserv2;             /* reserved                  */
      unsigned char       format;              /* reserved                  */
      AP_UINT16           primary_rc;          /* primary return code       */
      AP_UINT32           secondary_rc;        /* secondary return code     */
      unsigned char       pu_name[8];          /* PU name                   */
      unsigned char       description[32];     /* resource description      */
      unsigned char       reserv1[16];         /* reserved                  */
      unsigned char       reserv3[16];         /* reserved                  */
} DEFINE_DEFAULT_PU;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
デフォルトの設定 （ _PU ）

プール名
デフォルト PU の名前。これは、前の DEFINE_LS verb によって定義された PU 名でなければなりません。 これは、 8 バイトのタイプ A の EBCDIC ストリング (文字で始まる) で、必要に応じて右側に EBCDIC のスペースが埋め込まれます。

デフォルト PU を削除するには、すべてゼロを指定します。

記述
PU を記述したヌルで終了するテキスト・ストリング (0 から 31 文字の後にヌル文字を続けたストリング)。 このストリングは情報専用です。このストリングはノードの構成ファイルに保管され、 QUERY_DEFAULT_PU verb で戻されますが、 CS Linux ではそれ以外の使用は行われません。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

DEFINE_DEFAULTS

DEFINE_DEFAULTS specifies default parameters used by the node.

VCB 構造体

```
typedef struct define_defaults
{
      AP_UINT16           opcode;             /* verb operation code         */
      unsigned char       reserv2;            /* reserved                    */
      unsigned char       format;             /* reserved                    */
      AP_UINT16           primary_rc;         /* primary return code         */
      AP_UINT32           secondary_rc;       /* secondary return code       */
      DEFAULT_CHARS       default_chars;      /* default parameters          */
} DEFINE_DEFAULTS;
```

```
typedef struct default_chars
{
      unsigned char       description[32];    /* resource description        */
      unsigned char       reserv2[16];        /* reserved                    */
      unsigned char       mode_name[8];       /* default mode name           */
      unsigned char       implicit_plu_forbidden; /* disallow implicit PLUs? */
      unsigned char       specific_security_codes;/* generic security sense  */
                                              /* codes?                      */
      AP_UINT16           limited_timeout;    /* timeout for limited sessions*/
      unsigned char       reserv[244];        /* reserved                    */
} DEFAULT_CHARS;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
デフォルト値の追加 （ _C ）

default_chars.description
デフォルト・パラメーターを説明するヌルで終了するテキスト・ストリング (0 から 31 文字の後にヌル文字を続けたストリング)。 このストリングは情報専用であり、ノードの構成ファイルに保管され、 QUERY_DEFAULTS verb で戻されますが、 CS Linux ではそれ以外の使用は行われません。

デフォルト t_chars.mode_name
デフォルト・モードの名前。 セッションを開始しようとするときに、認識されないモード名をアプリケーションが指定すると、このモードからのパラメーターが、認識されないモードのデフォルト定義として使用されます。

これは、前の DEFINE_MODE verb で定義されたモードか、または 1 ページの『Purpose of the NOF API』 にリストされている SNA 定義モードのいずれかでなければなりません。 この名前は、 8 バイトのタイプ A の EBCDIC ストリング (文字で始まる) で、必要に応じて右側に EBCDIC のスペースが埋め込まれます。

***default_chars.implicit_plu_*禁じられた暗黙**
CS Linux が不明なパートナー LU に代わって暗黙定義を配置するかどうかを指定します。 可能な値は次のとおりです

類人猿
CS Linux は、不明なパートナー LU の代わりに暗黙定義を配置しません。 すべてのパートナー LU が明示的に定義されている必要

アプ・ノー
CS Linux は、不明なパートナー LU に代わって暗黙定義を代わりに使用します。

デフォルトの文字数 . 特殊セキュリティー・コード
CS Linux がセキュリティー認証または許可の失敗で特定のセンス・コードを使用するかどうかを指定します 特定のセンス・コードが戻されるのは、セッションでそれらのパートナー LU に対してサポートを報告しているパートナー LU のみです。 可能な値は次のとおりです

類人猿
CS Linux では特定のセンス・コードを使用

アプ・ノー
CS Linux は特定のセンス・コードを使用しません。

default_chars.limited_timeout
フリーの限定リソース・コン勝者セッションが非活動化されるまでのタイムアウトを指定します。 0 から 65,535 秒の範囲の値を指定します。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc

ファイル・パス名
モード名 パラメーターが定義済みのどのモード名とも一致しませんでした

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

定義ディレクトリー・エントリー

DEFINE_DIRECTORY_ENTRY は、ノード・ディレクトリー・データベース内に新しい項目を定義します。 この verb を使用して既存のエントリーを変更することはできませ この verb は、ネットワーク修飾リソース名をリソース・タイプ (ネットワーク・ノード、エンド・ノード、 LU 、またはワイルドカード) と一緒に提供します。

隣接ノードとその LU を定義する場合は、 DEFINE_DIRECTORY_ENTRY の代わりに DEFINE_ADJACENT_LEN_NODE を使用することをお勧めします。 これにより、単一 verb を使用してノードとその LU を定義することができます。 (DEFINE_DIRECTORY_ENTRY は単一の項目のみを定義します。したがって、隣接ノードとその LU の項目を定義するために複数の verb を使用する必要があります。)

データベースは階層構造であるため、各エントリーには親リソースの名前が含まれます。 LU の場合、親リソースは所有制御点であり、エンド・ノードまたは LEN ノードの場合はネットワーク・ノード・サーバーになります。 ただし、エンド・ノードまたは LEN ノードで DEFINE_DIRECTORY_ENTRY が使用されて、直接通信する隣接 LEN ノード・リソースを定義する場合、そのエントリーには親リソース名は含まれません。

名前の最初の文字だけを指定することによって、複数の LU 名に一致するように "ワイルドカード" LU 名を指定することができます。 例えば、ワイルドカード LU 名 APPN.LU は APPN.LUNAME または APPN.LU01 と一致します (ただし、 APPN.NAMELU とは一致しません)。

VCB 構造体

```
typedef struct define_directory_entry
{
   AP_UINT16        opcode;              /* verb operation code              */
   unsigned char    reserv2;             /* reserved                         */
   unsigned char    format;              /* reserved                         */
   AP_UINT16        primary_rc;          /* primary return code              */
   AP_UINT32        secondary_rc;        /* secondary return code            */
   unsigned char    resource_name[17];   /* network qualified resource name  */
   unsigned char    reserv1a;            /* reserved                         */
   AP_UINT16        resource_type;       /* resource type                    */
   unsigned char    description[32];     /* resource description             */
   unsigned char    reserv3[16];         /* reserved                         */
   unsigned char    parent_name[17];     /* fully qualified parent name      */
   unsigned char    reserv1b;            /* reserved                         */
   AP_UINT16        parent_type;         /* parent's resource type           */
   unsigned char    reserv4[8];          /* reserved                         */
} DEFINE_DIRECTORY_ENTRY;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
追加のディレクトリー・ エントリーの追加

リソース名
登録されるリソースの完全修飾名。 この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

リソース・タイプ
定義するリソースのタイプを指定します。 可能な値は次のとおりです

AP_ENCP_RESOURCE
エンド・ノードまたは LEN ノード

リソースの追加
ネットワーク・ノード

AP_LU_RESOURCE
ルゥ

追加のワイルドカード・リソース
ワイルドカードの LU 名。

LU またはワイルドカード LU の場合、親リソース (所有する CP) のディレクトリー項目は、すでに定義されていなければなりません。

記述
ディレクトリー項目を記述するヌルで終了するテキスト・ストリング (0 から 31 文字の後にヌル文字を続けたストリング) 。 このストリングは情報専用です。このストリングはノードの構成ファイルに保管され、 QUERY_DIRECTORY_ENTRY verb および QUERY_DIRECTORY_LU verb で戻されますが、 CS Linux ではそれ以外の使用は行われません。

parent_name
親リソースの完全修飾名。 LU の場合、親リソースは所有制御点であり、エンド・ノードまたは LEN ノードの場合はネットワーク・ノード・サーバーになります。 この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

このパラメーターは、以下の場合にはすべて 2 進ゼロに設定する必要があります。

- ネットワーク・ノード CP を登録する場合
- ローカル・ノードが直接通信する隣接 LEN ノード CP を定義するために、エンド・ノードまたは LEN ノードに対して verb が発行される場合。

parent_type
定義するリソースの親タイプを指定します。 可能な値は次のとおりです

AP_ENCP_RESOURCE
エンド・ノード (エンド・ノードによって所有される LU リソースの場合)

リソースの追加
ネットワーク・ノード (ネットワーク・ノードが所有する LU リソースの場合、または EN リソースまたは LEN リソースの場合) 。

親リソース名を指定しない場合は、このパラメーターをゼロに設定します。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

FQINVALID_FQ_OWNING_CP_NAME
parent_name パラメーターが、定義されたリソースの名前と一致しませんでした。

ファイル名の変更
リソース名 パラメーターに、無効な文字が含まれているか、または正しい形式ではありませんでした。

ソース・リソース・タイプのタイプ
リソース・タイプ パラメーターが有効な値に設定されていませんでした。

ファイル・ワイルドカード名を使用できます
リソース・タイプ パラメーターが **追加のワイルドカード・リソース**に設定されましたが、 リソース名 パラメーターに有効なワイルドカード・エントリーが含まれていませんでした。

重複
リソース名 パラメーターに、すでに定義されているワイルドカード項目が含まれていました。

ソース・リソース名が無効です
リソース名 パラメーターは、 verb が発行されたノードの名前と競合するノード名を指定しました。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

DEFINE_DLC

DEFINE_DLC defines a new DLC. It can also be used to modify the DLC-specific parameters of an existing DLC, if the DLC is not currently active, but other parameters (such as DLC type, negotiable link support and the valid port types) cannot be modified.

VCB structure

```
typedef struct define_dlc
{
   AP_UINT16       opcode;              /* verb operation code              */
   unsigned char   reserv2;             /* reserved                         */
   unsigned char   format;              /* reserved                         */
   AP_UINT16       primary_rc;          /* primary return code              */
   AP_UINT32       secondary_rc;        /* secondary return code            */
   unsigned char   dlc_name[8];         /* name of DLC                      */
   DLC_DEF_DATA    def_data;            /* DLC defined data                 */
} DEFINE_DLC;
```

```
typedef struct dlc_def_data
{
   unsigned char   description[32];      /* resource description            */
   unsigned char   initially_active;     /* is the DLC initially active?    */
   unsigned char   reserv1[15];          /* reserved                        */
   unsigned char   dlc_type;             /* DLC type                        */
   unsigned char   neg_ls_supp;          /* negotiable link station support */
   unsigned char   port_types;           /* port types supported by DLC type */
   unsigned char   hpr_only;             /* only support HPR?               */
   unsigned char   reserv3;              /* reserved                        */
   unsigned char   retry_flags;          /* reserved                        */
   AP_UINT16       max_activation_attempts; /* reserved                     */
   AP_UINT16       activation_delay_timer;  /* reserved                     */
   unsigned char   reserv4[4];              /* reserved                     */
   AP_UINT16       dlc_spec_data_len;       /* Length of DLC specific data  */
} DLC_DEF_DATA;
```

DLC-specific data for multipath channel (MPC), CS Linux for IBM Z only:

```
typedef struct chnl_dlc_spec_data
{
   V0_MUX_INFO     mux_info;             /* streams information            */
   AP_UINT16       mu_credit;            /* reserved                       */
   unsigned char   stats_support;        /* reserved                       */
   unsigned char   reserve1[31];         /* pad and future expansion       */
} CHNL_DLC_SPEC_DATA;
```

DLC-specific data for Enterprise Extender (HPR/IP):

```
typedef struct ipdlc_dlc_spec_data
{
    V0_MUX_INFO   mux_info;          /* streams information                */
    AP_UINT16     udp_port[5];       /* UDP port numbers for traffic       */
                                     /*  priorities LLC, Network, High,    */
                                     /*  Medium, Low                       */
    unsigned char ip_precedence[5];  /* IP precedence 0-7 for traffic      */
```

```
                                    /*  priorities                          */
    unsigned char no_dns_lookup;    /* are all remote hosts specified by    */
                                    /*  numeric IP address?                 */
} IPDLC_DLC_SPEC_DATA;
```

DLC-specific data for SDLC:

```
typedef struct sdl_spec_data
{
   V0_MUX_INFO       mux_info;        /* Streams config info                */
   AP_UINT16         mu_credit;       /* amount of credit to allow PC to send*/
   unsigned char     stats_support;   /* activate statistics gathering?     */
   unsigned char     reserve1;        /* reserved                           */
   AP_UINT16         sdh_parms_len;   /* Length of HMOD stub create_parms   */
   SDH_CREATE_PARMS  sdh_parms;       /* HMOD stub create_parms structure   */
} SDL_SPEC_DATA;
typedef struct sdh_create_parms
{
   AP_UINT16         length;          /* Length of HMOD stub create_parms   */
   AP_UINT16         num_ports;       /* max number of ports DLC can support */
   AP_UINT32         creators_pid;    /* process ID of DLC                  */
   V0_MUX_INFO       mux_info;        /* reserved                           */
} SDH_CREATE_PARMS;
```

DLC-specific data for QLLC:

```
typedef struct vql_dlc_spec_data
{
   V0_MUX_INFO       mux_info;        /* streams config info                */
} VQL_DLC_SPEC_DATA;
```

DLC-specific data for Token Ring, Ethernet:

```
typedef struct vmc_dlc_cfg
{
   V0_MUX_INFO       mux_info;              /* Streams config info         */
   AP_UINT16         lan_type;              /* type of LAN                 */
   AP_UINT16         min_rcv_dsf;           /* reserved                    */
   unsigned char     device_name[32];       /* device name                 */
} VMC_DLC_CFG;
```

For all DLC types:

```
typedef struct v0_mux_info
{
   AP_UINT16         dlc_type;              /* DLC implementation type     */
   unsigned char     need_vrfy_fixup;       /* reserved                    */
   unsigned char     num_mux_ids;           /* reserved                    */
   AP_UINT32         card_type;             /* type of adapter card        */
   AP_UINT32         adapter_number;        /* DLC adapter number          */
   AP_UINT32         oem_data_length;       /* reserved                    */
   AP_INT32          mux_ids[5];            /* reserved                    */
} V0_MUX_INFO;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
非定義の DLC

dlc_name
DLC の名前。 これは 8 バイトの ASCII ストリングで、ローカルで表示可能な文字を使用し、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。

データの説明の説明
DLC を記述するヌルで終了するテキスト・ストリング (0 から 31 文字の後にヌル文字を続けたストリング)。 このストリングは情報専用です。このストリングはノードの構成ファイルに保管され、QUERY_DLC verb で戻されますが、 CS Linux ではそれ以外の使用は行われません。

def_data.initially_active
ノードの開始時にこの DLC が自動開始されるかどうかを指定します 可能な値は次のとおりです

類人猿
DLC は、ノードが開始されると自動的に開始されます。

アプ・ノー
DLC は、それを使用するポートまたは LS が最初は活動状態であると定義されている場合にのみ、自動的に開始されます。それ以外の場合は、手動で開始する

データ *.dlc_type* の定義
DLC のタイプ。 既存の DLC に対してこのパラメーターを変更することはできません。このパラメーターは、新規 DLC の作成時にのみ指定できます。 可能な値は次のとおりです

アプスドゥルク
シュルツ

エイプックス 25
クルック

属性の追加
トークンリング

エイプのイーサネット
イーサネット

アプンプック
マルチパス・チャネル (MPC)、 CS Linux for IBM Z のみ

アプアップ
エンタープライズ・エクステンダー (HPR/IP)

データを無視します。ネガールのサポート
DLC がネゴシエーション可能リンク・ステーションをサポートするかどうかを指定 既存の DLC に対してこのパラメーターを変更することはできません。このパラメーターは、新規 DLC の作成時にのみ指定できます。 *dlc_type* が **エイプ・クルズ**に設定されている場合、これは **類人猿**に設定する必要があります。 *dlc_type* が **アプンプック**に設定されている場合、これは **類人猿**に設定する必要があります。

可能な値は次のとおりです

類人猿
この DLC を使用するリンク・ステーションは折衝可能です。

アプ・ノー
この DLC を使用するリンク・ステーションは、 1 次または 2 次のいずれかで定義する必要があります。折衝可能リンク・ステーションは

def_data.port_types
dlc_type が **属性の追加 / エイプのイーサネット / アプアップ**に設定されている場合は、このパラメーターを **ポートの追加** （ _SATF ）に設定します。 *dlc_type* が **アプンプック**に設定されている場合、このパラメーターを **ポート切り替え** （ _R ）に設定します。 その他のすべての DLC タイプの場合、このパラメーターは予約済みです。

データが不足しています。 *hpr_only*
DLC が HPR トラフィックのみをサポートするかどうかを指定 *dlc_type* が **アプアップ**に設定されている場合、これは **類人猿**に設定する必要があります。 *dlc_type* が **アプンプック**に設定されている場合、これは **アプ・ノー**に設定する必要があります。可能な値は次のとおりです

類人猿
この DLC は、 Enterprise Extender リンクに使用され、 HPR トラフィックのみをサポートします。

アプ・ノー
この DLC は、 Enterprise Extender 以外のリンク・タイプに使用され、非 HPR トラフィックをサポートします。 HPR トラフィックをサポートする場合もあります。

データのデフラグを定義してください。 *dlc_spec_data_len*
DLC のタイプに固有のデータの長さ(バイト単位) 。 DLC 固有のデータ構造は、基本 VCB 構造体の終わりに組み込む必要があります。

エンタープライズ・エクステンダー (HPR/IP)の DLC 固有データ :

ipdlc_dlc_spec_data.mux_info.dlc_type
DLC のタイプ。 これを **アプアップ**に設定する。

ipdlc_dlc_spec_data.mux_info.card_type
アダプター・カードのタイプ。 これを IP **の追加**に設定する。

ipdlc_dlc_spec_data.mux_info.adapter_number
予約済み (このパラメーターをゼロに設定) 。

ipdlc_dlc_spec_data.data.udp_port
DLC によって異なるトラフィック優先順位のために使用される 5 つの UDP ポート番号の配列。 これらは通常 12000 - 12004 に設定されます。

udp_port[0]
LLC コマンドに使用される UDP ポート。

udp_port[1]
ネットワーク優先順位トラフィックに使用される UDP ポート。

udp_port[2]
高優先度トラフィックに使用される UDP ポート。

udp_port[3]
中優先度トラフィックに使用される UDP ポート。

udp_port[4]
優先順位が低いトラフィックに使用される UDP ポート。

ipdlc_dlc_spec_data.data.ip_優先順位
DLC によって異なるトラフィック優先順位で使用される 5 つの IP 優先順位値の配列。 この配列内の各エントリーは、 0 (最小) - 7 (最大) の範囲の値です。

ip_優先順位 [0]
LLC コマンドに使用される IP 優先順位。 これは通常、 6 に設定されます

ip_優先順位 [1]
ネットワーク優先順位トラフィックに使用される IP 優先順位。 これは通常、 6 に設定されます

ip_優先順位 [2]
高優先度トラフィックに使用される IP 優先順位。 これは通常、 4 に設定されます

ip_優先順位 [3]
中優先順位トラフィックに使用される IP 優先順位。 これは通常、 2 に設定されます

ip_優先順位 [4]
低優先度トラフィックに使用される IP 優先順位。 これは通常、 1 に設定されます

ipdlc_dlc_spec_data.no_dns_lookup
リモート・ホスト IP アドレスがドメイン・ネーム・サーバーで検索を必要とするかどうかを 可能な値は次のとおりです

類人猿
着信 IP 接続の受信時にリモート IP アドレスからホスト名を検索しようとしないでください。

リモート IP アドレスを解決できない場合は、このオプションを使用します。 この場合、着信接続は、 LS がホスト名ではなく明示的な IP アドレス (IPv4 または IPv6 のいずれか) を使用するように構成されている場合にのみ、構成済み LS と一致させることができます。

アプ・ノー
この DLC に定義されたリンク・ステーションのリモート・ホスト IP アドレスは、名前 (newbox.this.co.uk など) または別名 (newbox など) として、数値アドレス (IPv4 または IPv6 のいずれか) として指定することができます。 ノードは、ドメイン・ネーム・サーバー・ルックアップを実行して、必要な場合にすべての着信呼び出しでリモート・ホスト名を判別します。

MPC 用の DLC 固有データ、 CS Linux for IBM Z のみ:

chnl_dlc_spec_data.mux_info.dlc_type
DLC のタイプ。 これを **実装の** MPC_GDLC に設定する。

chnl_dlc_spec_data.mux_info.card_type
アダプター・カードのタイプ。 これを **追加** IBM_ESCON に設定する。

chnl_dlc_spec_data.mux_info.adapter_number
このパラメーターは予約済みです (ゼロに設定します)。

SDLC の DLC 固有データ :

sdl_spec_data.mux_info.dlc_type
DLC のタイプ。 これを **追加可能** SDLC_SL に設定する

sdl_spec_data.mux_info.card_type
アダプター・カードのタイプ。

可能な値は次のとおりです

- **追加された** IBM_SDLC
- **付加された** IBM_MPCA
- **追加** IBM_MPAA
- **追加された文字の追加** （ _C ）

sdl_spec_data.mux_info.adapter_number
DLC によって使用されるアダプター 番号。 サーバーに複数の SDLC アダプター・カードが含まれている場合は、最初のカードに 0 (ゼロ) を指定し、 2 番目のカードの場合は 1 を指定し、以下のように指定します。 それ以外の場合、このパラメーターを 0 (ゼロ)に設定します

sdl_spec_data.mu_クレジット
DLC_MU をリンク・コンポーネントに送信するためのクレジット値を指定します。 このパラメーターを 4 に設定します。

sdl_spec_data.stats_support
DLC がリンク統計情報を収集するかどうかを指定 可能な値は次のとおりです

類人猿
この DLC は、 QUERY_STATISTICS を使用して調べることができるリンク統計情報を収集します。

アプ・ノー
DLC は、リンク統計情報を収集しません。

sdl_spec_data.sdh_parms_len
sdh_parms 構造体の長さ(バイト単位)。 これを sizeof(SDH_CREATE_PARMS)に設定する。

sdl_spec_data.sdh_parms.length
sdh_parms 構造体の長さ(バイト単位)。 これを sizeof(SDH_CREATE_PARMS)に設定する。

sdl_spec_data.sdh_parms.num_port
この DLC が一度にサポートする必要のあるポートの最大数。

sdl_spec_data.sdh_parms.creators_pid
このパラメーターは予約済みです (ゼロに設定します)。

QLLC の DLC 固有データ :

vql_dlc_spec_data.mux_info.dlc_type
DLC のタイプ。 これを **実装** NIMPL_QLLC に設定する。

vql_dlc_spec_data.mux_info.card_type
アダプター・カードのタイプ。 これを **追加された** QLLC_NLI に設定する。

vql_dlc_spec_data.mux_info.adapter_number
DLC によって使用されるアダプター 番号。 サーバーに複数の X.25 アダプター・カードが含まれている場合は、最初のカードにはゼロを指定し、 2 番目のカードには 1 を指定してください。 それ以外の場合は、このパラメーターをゼロに設定

トークンリング、イーサネットの DLC 固有データ

vmc_dlc_cfg.mux_info.dlc_type
DLC のタイプ。 可能な値は次のとおりです

暗黙の追加移植数 （ _LLC2 ）
トークンリング

暗黙のインザッスム・スナップショット _LLC2
イーサネット

vmc_dlc_cfg.mux_info.card_type
アダプター・カードのタイプ。

可能な値は次のとおりです

追加トークン・リングリング （ _L ）
トークンリング

追加のイーサネット・タスクの数
イーサネット

vmc_dlc_cfg.mux_info.adapter_number
DLC によって使用されるアダプター 番号。

この DLC タイプに対して複数のアダプター・カードがサーバーに含まれている場合は、最初のカードにゼロを指定し、 2 番目のカードには 1 を指定し、以下同様にします。 それ以外の場合は、このパラメーターをゼロに設定

vmc_dlc_cfg.lan_type
DLC によってアクセスされるネットワークのタイプ。 可能な値は次のとおりです

LLC_DIX
ディックス

LLC2_802_3
802.3

LLC2_802_3_DIX
判別されない (802.3 または DIX のいずれか) 。 CS Linux は、隣接局がこれらのフォーマットのうちの 1 つのフレームに最初に応答するときに、正しいタイプ (上記の 2 つの値の 1 つ) を検出します。

LLC2_TOKEN_RING
トークンリング

vmc_dlc_cfg.device_name
このパラメーターは、イーサネット DLC にのみ使用されます。 これが NULL でない場合は、adapter_number を使用する代わりにデバイス名を指定します。 これにより、 **エンス** 32 (RHEL システム上のデフォルト名) は、名前の代わりに使用されます `eth0`、 `eth1` **など** のような名前が

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

ファイル名の変更ができない
指定された *dlc_name* パラメーターに、無効な文字が含まれていました。

ファイル・タイプの追加タイプ
指定された *dlc_type* パラメーターは、許可された値ではありませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc
可能な値は次のとおりです

追加 DLC_アクティブ
指定された DLC は、現在活動状態であるため、変更できません。

ファイル・タイプの追加タイプ
既存の DLC の DLC タイプ、折衝可能リンク・サポート、またはサポートされるポート・タイプを変更することはできません。 これらは、新規 DLC の作成時にのみ指定できます。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **状態検査の追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

DEFINE_DLUR_DEFAULTS

DEFINE_DLUR_DEFAULTS defines a default Dependent LU server (DLUS) and a backup default DLUS; if a default DLUS or backup default DLUS is already defined, the verb overrides the existing definition. The default DLUS name is used by DLUR 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 a downstream PU.)

The verb can also be used to revoke a default DLUS or backup default DLUS, so that none is defined.

VCB structure

```
typedef struct define_dlur_defaults
{
  AP_UINT16       opcode;                 /* verb operation code          */
  unsigned char   reserv2;                /* reserved                     */
  unsigned char   format;                 /* reserved                     */
  AP_UINT16       primary_rc;             /* primary return code          */
  AP_UINT32       secondary_rc;           /* secondary return code        */
  unsigned char   description[32];        /* resource description         */
  unsigned char   reserv1[16];            /* reserved                     */
  unsigned char   dlus_name[17];          /* DLUS name                    */
  unsigned char   bkup_dlus_name[17];     /* Backup DLUS name             */
  unsigned char   reserv3;                /* reserved                     */
  unsigned char   dlus_retry_timeout;     /* retry timeout                */
  unsigned char   dlus_retry_limit;       /* retry limit                  */
  unsigned char   prefer_active_dlus;     /* reserved                     */
  unsigned char   persistent_pipe_support;/* reserved                     */
  unsigned char   reserv4[14];            /* reserved                     */
} DEFINE_DLUR_DEFAULTS;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
デフォルトの追加 DLUR_DEFAULTS

記述
DLUR のデフォルトを記述するヌルで終了するテキスト・ストリング (0 から 31 文字の後にヌル文字が続く)。 このストリングは情報専用であり、ノードの構成ファイルに保管されていますが、 CS Linux ではそれ以外の使用は行われません。

dlus_name
デフォルトとして使用される DLUS ノードの名前。 この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

現在のデフォルト DLUS を取り消して、デフォルトの DLUS が定義されないようにするには、このパラメーターを 17 の 2 進ゼロに設定します。

bkup_dlus_name
バックアップ・デフォルトとして使用される DLUS ノードの名前。 この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

現行のバックアップ・デフォルト DLUS を取り消して、バックアップ・デフォルト DLUS が定義されないようにするには、このパラメーターを 17 の 2 進ゼロに設定します。

dlus_retry_timeout
DLUS に接続するための再活動化タイマー。 CS Linux が DLUS に接続できない場合、このパラメーターは再試行間の時間を秒単位で指定します。 0x0001-0xFFFF の範囲の値を指定してください。

dlus_retry_limit
DLUS に連絡するための再試行カウント。 このパラメーターは、 CS Linux が最初の試みで DLUS との接続に失敗した場合に再試行する回数を指定するために使用されます。

0x0001-0xFFFE、または 0xFFFF の範囲内で値を指定して、 CS Linux が DLUS に接続するまで無期限に再試行することを指定します。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

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

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: 関数はサポートされません

ローカル・ノードの構成がそれをサポートしていないために verb が正常に実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc

付加機能がサポートされていません
ローカル・ノードは DLUR をサポートしていません。これは、 DEFINE_NODE verb の *dlur_support* パラメーターによって定義されます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

定義ファイルの定義

DEFINE_DOMAIN_CONFIG_FILE は、ドメイン構成ファイルのヘッダーに組み込まれるコメント・ストリングを指定するか、既存のコメント・ストリングを変更します。

ノード構成ファイルに相当する動詞はありません。このファイルのヘッダーにはコメント・ストリングが含まれていないため、 DEFINE_NODE verb の記述パラメーターを使用して、ノード構成ファイルにコメント情報を組み込みます。

この verb は、ドメイン構成ファイルに対して発行する必要があります。

VCB structure

```
typedef struct define_domain_config_file
{
   AP_UINT16            opcode;                 /* verb operation code       */
   unsigned char        reserv2;                /* reserved                  */
   unsigned char        format;                 /* reserved                  */
   AP_UINT16            primary_rc;             /* primary return code       */
   AP_UINT32            secondary_rc;           /* secondary return code     */
   unsigned char        reserv3[8];             /* Reserved                  */
   CONFIG_FILE_HEADER   hdr;                    /* defined data              */
} DEFINE_DOMAIN_CONFIG_FILE;
```

```
typedef struct config_file_header
{
   AP_UINT16            major_version;          /* reserved                  */
   AP_UINT16            minor_version;          /* reserved                  */
   AP_UINT16            update_release;         /* reserved                  */
   AP_UINT32            revision_level;         /* reserved                  */
   unsigned char        comment[100];           /* optional comment string   */
   unsigned char        updating;               /* reserved                  */
} CONFIG_FILE_HEADER;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DEFINE_DOMAIN_CONFIG_FILE

hdr.comment
An optional comment string to store information about the file. This is an ASCII string of 0-99 characters, followed by a null character. This parameter is for information only; CS Linux returns it on the QUERY_DOMAIN_CONFIG_FILE verb, but does not make any other use of it.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

定義の簡素化 (LU)

DEFINE_DOWNSTREAM_LU は、新しいダウンストリーム LU を定義し、それをアップストリーム・ホスト LU または LU プール (DEFINE_LU_0_TO_3 または DEFINE_LU_POOL を使用して定義される) にマップします。 これにより、 CS Linux の SNA ゲートウェイ機能を使用して、ダウンストリーム LU がホスト・コンピューターにアクセスできるようになります。 この verb は、既存のダウンストリーム LU の変更には使用できません。

この verb は、既に定義されているダウンストリーム LU を活動化するためにも使用できます (例えば、ダウンストリーム・ワークステーションが活動化されているためです)。 これを行うには、その LU に対して DEFINE_DOWNSTREAM_LU verb を再発行します。 定義を変更することはできないので、すべてのパラメーターは元の定義のものと同じでなければならないことに注意してください。

DEFINE_DOWNSTREAM_LU は、 CS Linux 基本 RUI アプリケーションと通信するアプリケーションによって使用されるダウンストリーム LU を定義する場合にも使用できます。 1 次 RUI の詳細については、「AIX または Linux LUA プログラマーズ・ガイド」の IBM Communications Server for Data Center デプロイメントを参照してください。

VCB structure

```
typedef struct define_downstream_lu
{
    AP_UINT16               opcode;           /* verb operation code       */
    unsigned char           reserv2;          /* reserved                  */
    unsigned char           format;           /* reserved                  */
    AP_UINT16               primary_rc;       /* primary return code       */
    AP_UINT32               secondary_rc;     /* secondary return code     */
    unsigned char           dslu_name[8];     /* Downstream LU name        */
    DOWNSTREAM_LU_DEF_DATA  def_data;         /* Defined data              */
} DEFINE_DOWNSTREAM_LU;
```

```
typedef struct downstream_lu_def_data
    unsigned char           description[32];   /* resource description      */
    unsigned char           reserv1[16];      /* reserved                  */
    unsigned char           nau_address;      /* downstream LU nau address */
    unsigned char           dspu_name[8];     /* Downstream PU name        */
    unsigned char           host_lu_name[8];  /* Host LU or Pool name      */
    unsigned char           allow_timeout;    /* Allow timeout of host LU? */
    unsigned char           delayed_logon;    /* Allow delayed logon to    */
                                              /*  host LU                  */
    unsigned char           reserv2[6];       /* reserved                  */
} DOWNSTREAM_LU_DEF_DATA;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DEFINE_DOWNSTREAM_LU

dslu_name
Name of the downstream LU that is being defined. This is an 8-byte type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces.

def_data.description
A null-terminated text string (0-31 characters followed by a null character) describing the downstream LU. This string is for information only; it is stored in the node's configuration file and returned on the QUERY_DOWNSTREAM_LU verb, but CS Linux does not make any other use of it.

def_data.nau_address
Network accessible unit address of the downstream LU. This must be in the range 1-255.

def_data.dspu_name
Name of the downstream PU associated with this LU (as specified on the DEFINE_LS). This is an 8-byte type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces.

def_data.host_lu_name
Name of the host LU or host LU pool that the downstream LU will be mapped onto. This is an 8-byte type-A EBCDIC string, padded on the right with EBCDIC spaces.

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 CS Linux Primary RUI application instead of a host, set this field to the string #PRIRUI# in EBCDIC.

def_data.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:

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

AP_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 CS Linux Primary RUI application instead of a host.

def_data.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:

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

AP_NO
Do not use delayed logon with this downstream LU.

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

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

アプインバリデータ・ファイル名
指定された *dslu_name* パラメーターに、無効な文字が含まれていました。

追加の無効メールアドレス
指定された NAU アドレスが有効範囲内にありませんでした。

無期限の ALLOW_TIMEOUT
指定された *allow_timeout* パラメーター値が無効でした。

追加のログオンを使用するログオン
指定された ログオンの *delayed_logon* パラメーター値が無効でした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: state check

If the verb does not execute because of a state error, CS Linux returns the following parameters.

primary_rc
AP_STATE_CHECK

secondary_rc
Possible values are:

AP_INVALID_PU_NAME
The specified *dspu_name* parameter was not valid.

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

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

AP_LU_ALREADY_DEFINED
An LU with the specified name has already been defined, and cannot be modified using this verb.

AP_DSLU_ACTIVE
The LU is already active.

AP_LU_NAU_ADDR_ALREADY_DEFD
An LU with the specified NAU address has already been defined.

AP_INVALID_HOST_LU_NAME
The specified host LU name was not valid.

AP_LU_NAME_POOL_NAME_CLASH
The specified LU name clashes with the name of an existing LU pool.

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

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

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

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_STATE_CHECK, which are common to all NOF verbs.

Returned parameters: function not supported

If the verb does not execute successfully because the local node's configuration does not support it, CS Linux returns the following parameters:

primary_rc

AP_FUNCTION_NOT_SUPPORTED
The local node does not support SNA gateway; this is defined by the *pu_conc_support* parameter on the DEFINE_NODE verb.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

実行範囲が定義された定義の数

DEFINE_DOWNSTREAM_LU_RANGE は、ダウンストリーム LU の新しい範囲を定義し、それらをアップストリーム・ホスト LU または LU プール (DEFINE_LU_0_TO_3 または DEFINE_LU_POOL を使用して定義) にマップします。 これにより、ダウンストリーム LU は CS Linux の SNA ゲートウェイ機能を使用してホスト・コンピューターにアクセスすることができます。 この verb を使用して、既存のダウンストリーム LU を変更することは

この verb に提供されるパラメーターには、新しい LU のベース名と NAU アドレスの範囲が含まれます。 新しい LU 名は、ベース名と NAU アドレスを結合することによって生成されます。 例えば、 LUNME のベース名と NAU 範囲の 11 から 14 を組み合わせると、 LU LUNME011、 LUNME012、 LUNME013 、および LUNME014 が定義されます。

DEFINE_DOWNSTREAM_LU_RANGE を使用して、 CS Linux 基本 RUI アプリケーションと通信するアプリケーションが使用するダウンストリーム LU を定義することもできます。 1 次 RUI の詳細については、「 AIX または Linux LUA プログラマーズ・ガイド」の IBM Communications Server for Data Center デプロイメントを参照してください。

VCB structure

```
typedef struct define_downstream_lu_range
{
    AP_UINT16         opcode;              /* verb operation code            */
    unsigned char     reserv2;             /* reserved                       */
    unsigned char     format;              /* reserved                       */
    AP_UINT16         primary_rc;          /* primary return code            */
    AP_UINT32         secondary_rc;        /* secondary return code          */
    unsigned char     dslu_base_name[5];   /* Downstream LU base name        */
    unsigned char     reserv3;             /* reserved                       */
    unsigned char     description[32];     /* resource description           */
    unsigned char     reserv1[16];         /* reserved                       */
    unsigned char     min_nau;             /* Minimum NAU address in range   */
    unsigned char     max_nau;             /* Maximum NAU address in range   */
    unsigned char     dspu_name[8];        /* Downstream PU name             */
    unsigned char     host_lu_name[8];     /* Host LU or Pool name           */
    unsigned char     allow_timeout;       /* Allow timeout of host LU?      */
    unsigned char     delayed_logon;       /* Allow delayed logon to host LU */
    unsigned char     reserv4[6];          /* reserved                       */
} DEFINE_DOWNSTREAM_LU_RANGE;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DEFINE_DOWNSTREAM_LU_RANGE

dslu_base_name
Base name for the names of the new LUs. This is a 5-byte type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces if the base name is less than 5 characters. CS Linux generates the LU name for each LU by appending the 3-digit decimal value of the NAU address to this name.

description
A null-terminated text string (0-31 characters followed by a null character) describing the downstream LUs (the same string is used for each LU in the range). This string is for information only; it is stored in the node's configuration file and returned on the QUERY_DOWNSTREAM_LU verb, but CS Linux does not make any other use of it.

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 the DEFINE_LS verb) which the downstream LUs in this range will use. This is an 8-byte type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces if necessary.

host_lu_name
Name of host LU or host LU pool that the downstream LUs in the given range will be mapped to. This is an 8-byte type-A EBCDIC string, padded on the right with EBCDIC spaces if necessary.

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

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:

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

AP_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 CS 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:

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

AP_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 CS Linux Primary RUI application instead of a host.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

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

AP_INVALID_NAU_ADDRESS
One or more of the supplied NAU addresses was not in the valid range.

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

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: state check

If the verb does not execute because of a state error, CS Linux returns the following parameters.

primary_rc
AP_STATE_CHECK

secondary_rc
Possible values are:

AP_INVALID_PU_NAME
The specified *dspu_name* parameter was not valid.

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

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

AP_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 verb.

AP_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 verb.

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

AP_INVALID_HOST_LU_NAME
The specified host LU name was not valid.

AP_LU_NAME_POOL_NAME_CLASH
One of the LU names in the range clashes with the name of an existing LU pool.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_STATE_CHECK, which are common to all NOF verbs.

戻りパラメーター: 関数はサポートされません

ローカル・ノードの構成がそれをサポートしていないために verb が正常に実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc

付加機能がサポートされていません
ローカル・ノードは SNA ゲートウェイをサポートしていません。これは、 DEFINE_NODE verb の *pu_conc__* サポート パラメーターによって定義されます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

定義さ DSPU_TEMPLATE

DEFINE_DSPU_TEMPLATE verb は、 CS Linux SNA ゲートウェイ機能を使用するダウンストリーム LU のテンプレートを定義します。 このテンプレートは、ワークステーションが暗黙リンク (以前に定義されていないリンク) を介して接続されるときに、ダウンストリーム・ワークステーションのグループにダウンストリーム LU を定義するために使用されます

DEFINE_DSPU_TEMPLATE を使用して、 CS Linux ノード上の 1 次 RUI アプリケーションと通信するアプリケーションをサポートするダウンストリーム LU を定義することもできます。 1 次 RUI の詳細については、「 AIX または Linux LUA プログラマーズ・ガイド」の IBM Communications Server for Data Center デプロイメントを参照してください。

VCB structure

```
typedef struct define_dspu_template
{
   AP_UINT16               opcode;            /* verb operation code        */
   unsigned char           reserv3;           /* reserved                   */
   unsigned char           format;            /* reserved                   */
   AP_UINT16               primary_rc;        /* primary return code        */
   AP_UINT32               secondary_rc;      /* secondary return code      */
   unsigned char           template_name[8];  /* Name of template           */
   unsigned char           description[32];   /* resource description       */
   unsigned char           reserv2[16];       /* reserved                   */
   unsigned char           modify_template;   /* Modify existing template?  */
   unsigned char           reserv1[11];       /* reserved                   */
   AP_UINT16               max_instance;      /* Max active template        */
                                              /* instances                  */
   AP_UINT16               num_of_dslu_templates;/* number of DSLU templates*/
} DEFINE_DSPU_TEMPLATE;

typedef struct dslu_template
{
   unsigned char           min_nau;           /* Minimum NAU address in range*/
   unsigned char           max_nau;           /* Maximum NAU address in range*/
   unsigned char           allow_timeout;     /* Allow timeout of host LU?   */
   unsigned char           delayed_logon;     /* Allow delayed logon to host */
                                              /*  LU                         */
   unsigned char           reserv1[8];        /* reserved                    */
   unsigned char           host_lu[8];        /* Host LU or Pool name        */
} DSLU_TEMPLATE;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DEFINE_DSPU_TEMPLATE

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 verb.

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

AP_MODIFY_DSPU_TEMPLATE
If the named DSPU template does not exist, then it is created. If the named DSPU template does exist, then appended DSLU templates are added to the existing DSPU template.

AP_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 (zero) indicates no limit.

num_of_dslu_templates
The number of downstream LU (DSLU) templates being defined by this verb.

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.

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:

AP_YES
CS Linux is allowed to timeout host LUs used by this downstream LU.

AP_NO
CS 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 CS 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:

AP_YES
CS Linux delays connecting the downstream LU to the host LU. A simulated logon screen is sent to the downstream LU.

AP_NO
CS 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 CS Linux Primary RUI application instead of a host.

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 CS Linux Primary RUI application instead of a host, set this field to the string #PRIRUI# in EBCDIC.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

アプリケーション・テンプレート名の変更
テンプレート名 パラメーターに指定された名前が無効でした。

追加の無効メールアドレス
ミンナウ パラメーターまたは 最大値 パラメーターが無効でした。

ファイルへの API の無効範囲
ミンナウ パラメーターまたは 最大値 パラメーターで指定されたアドレスが、有効な範囲内にありませんでした。

ファイルの追加の値の範囲
dslu_template サブレコード内の 最大値 パラメーターを使用して ミンナウ パラメーターによって指定されたアドレスの範囲は、 テンプレート名 パラメーターによって指定されたテンプレート内の別の dslu_template サブレコードによって指定された範囲と競合します。

テンプレートの追加情報テンプレート
*d_of_dslu_*テンプレートの数 パラメーターに指定された値が有効範囲内にありませんでした。

無期限の ALLOW_TIMEOUT
allow_timeout パラメーターに指定された値が有効ではありませんでした。

追加のログオンを使用するログオン
ログオンの *delayed_logon* パラメーターに指定された値が有効ではありませんでした。

変更された変更テンプレート
変更テンプレート パラメーターに指定された値が有効ではありませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc
可能な値は次のとおりです

ファイル名の入力が無効です
指定された ホスト名 パラメーター値が無効でした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **状態検査の追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: function not supported

If the verb does not execute successfully because the local node's configuration does not support it, CS Linux returns the following parameters:

primary_rc

AP_FUNCTION_NOT_SUPPORTED
The local node does not support SNA gateway; this is defined by the *pu_conc_support* parameter on the DEFINE_NODE verb.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

定義ポイント・フォーカル・ポイント

DEFINE_FOCAL_POINT verb は、特定の管理サービス・カテゴリーのフォーカル・ポイントを指定します。 新しいフォーカル・ポイントが指定されると、 CS Linux は、 MS_CAPABILITIES 要求を送信することによって、指定されたフォーカル・ポイントと暗黙の 1 次フォーカル・ポイント関係を確立しようと試みます。

VCB structure

```
typedef struct define_focal_point
{
   AP_UINT16        opcode;             /* verb operation code              */
   unsigned char    reserv2;            /* reserved                         */
   unsigned char    format;             /* reserved                         */
   AP_UINT16        primary_rc;         /* primary return code              */
   AP_UINT32        secondary_rc;       /* secondary return code            */
   unsigned char    reserved;           /* reserved                         */
   unsigned char    ms_category[8];     /* management services category     */
   unsigned char    fp_fqcp_name[17];   /* Fully qualified focal            */
                                        /*  point cp name                   */
   unsigned char    ms_appl_name[8];    /* Focal point application name     */
   unsigned char    description[32];    /* resource description             */
   unsigned char    reserv1[16];        /* reserved                         */
   unsigned char    backup;             /* is focal point a backup          */
   unsigned char    reserv3[16];        /* reserved                         */
} DEFINE_FOCAL_POINT;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
ファイル· フォーカル· ポイントの追加

ms_category
「管理サービス」カテゴリー。 これは、 システム・ネットワーク体系: 管理サービス参照 の「 MS 規律-固有アプリケーション・プログラム」テーブルに指定されたカテゴリー名の 1 つである可能性があります (「参考文献」を参照) 、 EBCDIC スペース (0x40 バイト)で埋め込まれたカテゴリー名、またはユーザー定義のカテゴリー名のいずれかです。 ユーザー定義のカテゴリー名は、 8 バイトのタイプ 1134 の EBCDIC ストリングで、必要に応じて、 EBCDIC スペース (0x40 バイト) が埋め込まれます。

fp_fqcp_name
フォーカル・ポイントの完全修飾制御点名。 この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

ms_appl_name
フォーカル・ポイント・アプリケーション名。 これは通常、タイプ - 1134 文字を使用して EBCDIC ストリングです。あるいは、 システム・ネットワーク体系: 管理サービス参照 で指定された MS 規律固有アプリケーション・プログラムの 1 つ (「参考文献」を参照) のいずれかになります。 ストリングは 8 文字の長さでなければなりません。必要に応じて、右側に EBCDIC スペース文字 (0x40 バイト) を埋め込む必要があります。

記述
フォーカル・ポイントを記述するヌルで終了するテキスト・ストリング (0 から 31 文字の後にヌル文字が続く) 。 このストリングは情報専用であり、ノードの構成ファイルに保管され、 QUERY_FOCAL_POINT verb で戻されますが、 CS Linux ではそれ以外の使用は行われません。

バックアップ
指定されたアプリケーションがこのカテゴリーのメイン ・フォーカル・ポイントであるか、バックアップ・フォーカル・ポイントであるかを示します。 可能な値は次のとおりです

類人猿
バックアップ・フォーカル・ポイント (メイン・フォーカル・ポイントが使用できない場合にのみ使用されます)。

アプ・ノー
メイン・フォーカル・ポイント。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc

AP_OK
The focal point was defined as requested.

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

ファイル・カテゴリー名の変更
指定されたカテゴリー名に、無効な文字が含まれていました。

ファイル名の表示名
完全修飾名またはアプリケーション名が有効ではありませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: 関数はサポートされません

ローカル・ノード構成がそれをサポートしていないために verb が正常に実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc

付加機能がサポートされていません
ローカル・ノードは MS ネットワーク管理機能をサポートしていません。これは、 DEFINE_NODE verb の *mds_supported* パラメーターによって定義されます。

Returned parameters: replaced

If the verb does not execute successfully because it is followed by another verb specifying a different focal point, CS Linux returns the following parameters.

primary_rc

AP_REPLACED
Another DEFINE_FOCAL_POINT was issued to the same node while this verb was outstanding, specifying a different focal point for the same MS category. This verb was abandoned; the node will attempt to contact the focal point specified by the second verb.

Returned parameters: unsuccessful

If the verb does not execute successfully because the focal point relationship cannot be established, CS Linux returns the following parameters:

primary_rc
AP_UNSUCCESSFUL

secondary_rc
Possible values are:

AP_IMPLICIT_REQUEST_REJECTED
The specified focal point rejected the request.

AP_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.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

定義さ AL_PU

DEFINE_INTERNAL_PU verb は、 DLUR がサービスを提供するローカル・ノード上の PU を定義します。 (DLUR または SNA ゲートウェイによって処理されるダウンストリーム PU を定義するか、ホストに直接接続されているローカル PU を定義するには、 DEFINE_INTERNAL_PU の代わりに DEFINE_LS を使用します。)

VCB structure

```
typedef struct define_internal_pu
{
      AP_UINT16             opcode;              /* verb operation code     */
      unsigned char         reserv2;             /* reserved                */
      unsigned char         format;              /* reserved                */
      AP_UINT16             primary_rc;          /* primary return code     */
      AP_UINT32             secondary_rc;        /* secondary return code   */
      unsigned char         pu_name[8];          /* internal PU name        */
      INTERNAL_PU_DEF_DATA  def_data;            /* defined data            */
} DEFINE_INTERNAL_PU;
```

```
typedef struct internal_pu_def_data
{
  unsigned char    description[32];              /* resource description     */
  unsigned char    initially_active;            /* is PU initially active?  */
  unsigned char    reserv1[15];                 /* reserved                 */
  unsigned char    dlus_name[17];               /* DLUS name                */
  unsigned char    bkup_dlus_name[17];          /* backup DLUS name         */
  unsigned char    pu_id[4];                    /* PU identifier            */
  AP_UINT16        dlus_retry_timeout;          /* DLUS retry timeout       */
  AP_UINT16        dlus_retry_limit;            /* DLUS retry limit         */
  unsigned char   conventional_lu_compression; /* compression for LU 0-3?   */
  unsigned char    conventional_lu_cryptography; /* reserved                */
  unsigned char    pu_can_send_dddlu_offline;   /* does the PU send NMVT    */
                                                /* (power off) to host?     */
  unsigned char    reserv2[1];                  /* reserved                 */
} INTERNAL_PU_DEF_DATA;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DEFINE_INTERNAL_PU

pu_name
Name of the internal PU that is being defined. This is a type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces.

This name should match the PU name configured on the host. (CS 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.)

def_data.description
A null-terminated text string (0-31 characters followed by a null character) describing the internal PU. This string is for information only; it is stored in the node's configuration file and returned on the QUERY_DLUR_PU and QUERY_PU verbs, but CS Linux does not make any other use of it.

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

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

AP_NO
The PU is not automatically started; it must be started manually.

def_data.dlus_name
Name of DLUS node which DLUR will use when it initiates SSCP-PU activation. The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

To indicate that DLUR should use the global default DLUS, set this parameter to 17 binary zeros. In this case, you must also use DEFINE_DLUR_DEFAULTS to define the global default DLUS.

def_data.bkup_dlus_name
Name of DLUS node which will serve as the backup DLUS for this PU. The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

To indicate that DLUR should use the global backup default DLUS, set this parameter to 17 binary zeros. In this case, you must also use DEFINE_DLUR_DEFAULTS to define the global backup default DLUS.

def_data.pu_id
PU identifier. This is a 4-byte hexadecimal string, consisting of a block number (3 hexadecimal digits) and a node number (5 hexadecimal digits). The PU ID should match the *pu_id* configured at the host. (CS 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.)

def_data.dlus_retry_timeout
Reactivation timer for contacting a DLUS. If CS Linux fails to contact the DLUS, this parameter specifies the time in seconds between retries. The interval between the first and second attempts is always 1 second.

Specify a value in the range `0x0001-0xFFFF`. If zero is specified, then the defaults specified using the DEFINE_DLUR_DEFAULTS verb are used.

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

Specify a value in the range `0x0001-0xFFFE`, or specify `0xFFFF` to indicate that CS Linux should retry indefinitely until it contacts the DLUS.

def_data.conventional_lu_compression
Specifies whether data compression is requested for LU 0-3 sessions using this PU.

Possible values are:

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

AP_NO
Data compression should not be used for LU 0-3 sessions using this PU.

def_data.pu_can_send_dddlu_offline
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), CS 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:

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

AP_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 AP_NO.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

エイプインバリデータプール名
プール名 パラメーターに、無効な文字が含まれていました。

追加の無効 PU_ID
パブリッシュ *ID* パラメーターに、無効な文字が含まれていました。

ファイル名の入力が無効です
dlus_name パラメーターに、無効な文字が含まれているか、または正しい形式ではありませんでした。

アプインバリデータ・ プクル名
bkup_dlus_name パラメーターに、無効な文字が含まれているか、または正しい形式ではありませんでした。

Returned parameters: state check

If the verb does not execute because of a state error, CS Linux returns the following parameters.

primary_rc
AP_STATE_CHECK

secondary_rc

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

戻りパラメーター: 関数はサポートされません

ノードの構成がそれをサポートしていないために verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc

付加機能がサポートされていません
ノードは DLUR をサポートしていません。これは、 DEFINE_NODE verb の *dlur_support* パラメーターによって定義されます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

定義 LOCAL_LU

DEFINE_LOCAL_LU verb は、新しいローカル LU を定義します。 また、既存の LU (またはローカル・ノードの制御点に関連付けられたデフォルト LU) の接続ルーティング・データ、 無効化 パラメーター、または説明を変更するために使用することもできますが、他のパラメーターのいずれも変更することはできません。既存の LU を変更する場合は、他のすべてのパラメーターは、現在定義されている値に設定する必要があります。

VCB 構造体

```
typedef struct define_local_lu
{
   AP_UINT16            opcode;           /* verb operation code            */
   unsigned char        reserv2;          /* reserved                       */
   unsigned char        format;           /* reserved                       */
   AP_UINT16            primary_rc;       /* primary return code            */
   AP_UINT32            secondary_rc;     /* secondary return code          */
   unsigned char        lu_name[8];       /* local LU name                  */
   LOCAL_LU_DEF_DATA    def_data;         /* defined data                   */
} DEFINE_LOCAL_LU;
```

```
typedef struct local_lu_def_data
{
   unsigned char    description[32];       /* resource description          */
   unsigned char    reserv1;               /* reserved                      */
   unsigned char    security_list_name[14];   /* security access list name  */
   unsigned char    reserv3;               /* reserved                      */
   unsigned char    lu_alias[8];           /* local LU alias                */
   unsigned char    nau_address;           /* NAU address                   */
   unsigned char    syncpt_support;        /* is Syncpoint supported?       */
   AP_UINT16        lu_session_limit;      /* LU session limit              */
   unsigned char    default_pool;          /* is LU in the pool of default  */
                                           /*  LUs?                         */
   unsigned char    reserv2;               /* reserved                      */
   unsigned char    pu_name[8];            /* PU name                       */
   unsigned char    lu_attributes;         /* LU attributes                 */
   unsigned char    sscp_id[6];            /* SSCP ID                       */
   unsigned char    disable;               /* disable or enable local LU    */
   ROUTING_DATA     attach_routing_data;   /* routing data for incoming     */
                                           /* attaches                      */
   unsigned char    reserv6;               /* reserved                      */
   unsigned char    reserv4[7];            /* reserved                      */
   unsigned char    reserv5[16];           /* reserved                      */
} LOCAL_LU_DEF_DATA;
```

```
typedef struct routing_data
{
   unsigned char    sys_name[128];         /* Name of target system for TP  */
   AP_INT32         timeout;               /* timeout value in seconds      */
   unsigned char    back_level;            /* reserved                      */
   unsigned char    reserved[59];          /* reserved                      */
} ROUTING_DATA;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
使用中のファイルの追加

lu_name
ローカル LU の名前。 これは、 8 バイトのタイプ A の EBCDIC ストリング (文字で始まる) で、右側に EBCDIC のスペースが埋め込まれます。 これは、他のローカル LU 名、またはパートナー LU の完全修飾パートナー LU 名と一致しないようにする必要があります。

ローカル・ノードの制御点に関連するデフォルト LU の接続ルーティング・データまたは記述を変更するには、このパラメーターを 8 桁の 2 進ゼロに設定します。

データの説明の説明
ローカル LU を記述するヌルで終了するテキスト・ストリング (0 から 31 文字の後にヌル文字を続けたストリング)。 このストリングは情報専用です。このストリングはノードの構成ファイルに保管され、 QUERY_LOCAL_LU verb で戻されますが、 CS Linux ではそれ以外の使用は行われません。

データ . セキュリティー・リスト名
このローカル LU によって使用されるセキュリティー・アクセス・リストの名前 (DEFINE_SECURITY_ACCESS_LIST verb を使用して定義される)。 このパラメーターは、指定されたリストに指定されたユーザーだけが使用できるように、 LU を制限します。 任意のユーザーが LU を使用できるように指定するには、このパラメーターを 14 の 2 進ゼロに設定します。

def_data.lu_alias
ローカル LU の別名。 これは 8 バイトの ASCII ストリングで、ローカルで表示可能な文字を使用し、必要に応じて右側に 8 バイトまで埋め込まれます。

def_data.nau_address
LU のネットワーク・アクセス可能な装置アドレス。 LU が独立 LU である場合はゼロを指定し、 LU が従属 LU である場合は 1-255 の範囲内のアドレスを指定します。

***データ.syncpt_* サポートの定義**
LU が同期点機能をサポートするかどうかを指定する。 これを **類人猿** に設定するのは、標準の CS Linux 製品に加えて、 Sync Point Manager (SPM) と会話保護リソース・マネージャー (C-PRM) がある場合に限ります。 可能な値は次のとおりです

類人猿
同期点がサポートされます。

アプ・ノー
同期点はサポートされていません。

***データ . lu_session_limit* のデータを定義しています**
LU によってサポートされるセッションの最大合計数 (すべてのモードで)。

従属型 LU の場合は、これを 1 に設定する必要があります。 独立型 LU の場合は、制限なしにゼロを指定するか、または 1-65,535 の範囲の値を指定してください。 明示的な制限を指定する場合は、次の点に注意してください。

- LU が並列セッションのリモート LU と通信する場合、セッション限度には、 CNOS 折衝用の十分なセッションが含まれている必要があります。安全な最小値は 3、またはパートナー LU ごとに 2 つのセッションを追加することができます。
- LU セッション限度は、その LU が使用するすべてのモードのセッション限度の合計より大きくなければなりません。
- LU が全二重 APPC 会話によって使用される場合、各全二重会話には 2 つのセッションが必要です。

def_data.default_pool
LU がデフォルトの従属 LU のプール内にあるかどうかを指定します。 詳しくは、 103 ページの『Default LUs』を参照してください。 可能な値は次のとおりです

類人猿
LU は、デフォルト LU のプール内にあり、 LU 名を指定しないアプリケーションで使用できます。

アプ・ノー
LU がプール内にありません。

LU が独立 LU の場合、このパラメーターは予約されます。

***データ .pu_name* の定義**
この LU が使用する PU の名前で、 DEFINE_LS verb で指定されているとおりです。 このフィールドは、従属 LU によってのみ使用され、独立 LU には 8 進ゼロに設定する必要があります。 この名前は、8 バイトのタイプ A の EBCDIC ストリング (文字で始まる) で、必要に応じて右側に EBCDIC のスペースが埋め込まれます。

データの更新属性
LU に関する追加情報を識別します。 可能な値は次のとおりです

追加なし
追加情報はありません。

PWSUB を使用不可にする
ローカル LU のパスワード置換サポートを使用不可にします。 パスワード置換とは、平文として送信されるのではなく、ローカル LU とリモート LU の間でパスワードが暗号化されることを意味します。 CS Linux は、リモート・システムがパスワード置換をサポートしている場合、通常はパスワード

この値は、パスワード置換を正しく実装しないいくつかのリモート・システムとの通信のために、回避策として提供されます。 このオプションを使用する場合は、パスワードを平文で送信および受信する必要があることに注意する必要があります (セキュリティー・リスクを表す可能性があります)。 リモート・システムのパスワード置換の実装に問題がない限り、これを設定しないでください。

データ .sscp_id
この LU の活動化を許可された SSCP の ID を指定します。 この ID は 6 バイトのバイナリー・ストリングです。 このパラメーターは、従属 LU によってのみ使用され、 LU が独立 LU である場合、または LU がどの SSCP によっても活動化される場合には、すべて 2 進ゼロに設定されます。

def_data.disable
ローカル LU を使用不可にするか有効にするかを指定します。 このフィールドは、従属する LU6.2 LU にのみ使用され、それ以外の場合は予約済みです。 可能な値は次のとおりです

類人猿
ローカル LU を無効にします。

アプ・ノー
ローカル LU を有効にします。

def_data.attach_routing_data.sys_name
このローカル LU に到着する着信割り振り要求 (APPC または CPI-C 会話を開始するためのパートナー TP からの要求) のターゲット・コンピューターのシステム名。

ターゲット TP がブロードキャスト待機 TP である場合 (つまり、サーバーは始動時にそのロケーションを通知されるため、着信割り振り要求をそれに経路指定することができます)、またはこの LU を所有するノードと同じ CS Linux サーバー上で常に実行される場合は、このパラメーターを 2 進ゼロに設定します。 それ以外の場合は、 TP が実行されているコンピューターの名前に設定します。

この名前は、別名か完全修飾名のいずれかでなければなりません。 IP アドレスを指定することはできません。 システム名に . (ピリオド) 文字が含まれている場合、 CS Linux は、その名前が完全修飾名であると見なします。それ以外の場合は、 DNS ルックアップを実行してシステム名を判別します。

def_data.attach_routing_data.timeout
動的ロード要求のタイムアウト値。 呼び出された TP が、この時間内に Receive_Allocate verb (APPC)、または Accept_Conversation または Accept_Incoming (CPI-C)を発行していない場合は、要求はタイムアウトになります。 タイムアウト値 (秒単位) を指定するか、タイムアウトなし (動的ロード要求が無期限に待機する) を -1 します。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

使用不可にする
無効化 パラメーターが有効な値に設定されていませんでした。

ファイル名の変更
指定された LU 名に、無効な文字が含まれていました。

追加の無効メールアドレス
指定された NAU アドレスが有効範囲内にありませんでした。

追加の無効セッション制限
指定されたセッション限度が有効範囲内にありませんでした。

ファイル無効タイムアウト
指定されたタイムアウト値が有効範囲内にありませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc
可能な値は次のとおりです

ファイル名の変更
lu_name または *lu_alias* パラメーターのいずれかに、無効な文字が含まれているか、または LU 名がパートナー LU の完全修飾名と一致していました。

追加の ALREADY_DEFINED
この名前の LU はすでに定義されています。 この verb を使用して、接続ルーティング・データを除く既存の LU のパラメーターを変更することはできません。

付加が定義されていない
プール名 パラメーターが定義済みのどの PU 名とも一致しませんでした

追加のセキュリティー・ リストが定義されていない
セキュリティー・リスト名 パラメーターが、定義されたセキュリティー・アクセス・リスト名と一致しませんでした

使用されているエイリアス・ エイリアスが使用されています
この別名を持つ LU はすでに定義されています。 この verb を使用して、接続ルーティング・データを除く既存の LU のパラメーターを変更することはできません。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **状態検査の追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

Default LUs

You can set up the configuration of local LUs so that applications do not have to specify an LU name explicitly when starting a conversation; the node will select a suitable default LU for the application to use. The method for doing this depends on whether the applications require dependent or independent LUs, as follows. You cannot provide this facility for both dependent and independent LUs.

- If the applications require dependent LUs, use the *default_pool* parameter on DEFINE_LOCAL_LU for one or more dependent LUs, to specify that they can be used as default LUs. When an application

attempts to start a conversation without specifying a local LU name, CS Linux will select an unused LU from the pool of LUs defined as default LUs.

- You can define LUs on more than one node as default LUs. An application requesting a default LU may be assigned to any of these LUs as available; there is no requirement for the LU to be on the same computer as the application. However, if you are defining partner LUs for the applications, these must be defined on all nodes where default LUs are defined (so that the application can contact the correct partner LU using any of the default local LUs).
- If the applications require independent LUs, do not use the *default_pool* parameter to define any local LUs as default LUs. In this case, an application requesting a default LU will be assigned to the LU associated with a local node's CP (this is an independent LU automatically defined by CS Linux for each node).

DEFINE_LS

DEFINE_LS is used to define a new link station (LS) or modify an existing one. Before issuing this verb, you must issue the DEFINE_PORT verb to define the port that this LS uses. Link specific data is concatenated to the basic structure.

If you are defining a Multipath Channel (MPC) LS, only one active LS can be associated with each MPC port (which defines the MultiPath Channel device /dev/mpc*n*), because the port configuration includes the addressing information that identifies the host. If you need to support more than one active MPC LS at a time, you must define multiple ports, and define an LS for each of these ports. (Multipath Channel is available only with CS Linux for IBM Z.)

You cannot use DEFINE_LS to modify the port used by an existing LS; the *port_name* specified on the verb must match the previous definition of the LS. The LS can be modified only if it is not started.

VCB structure

```
typedef struct define_ls
{
   AP_UINT16       opcode;                 /* verb operation code            */
   unsigned char   reserv2;                /* reserved                       */
   unsigned char   format;                 /* reserved                       */
   AP_UINT16       primary_rc;             /* primary return code            */
   AP_UINT32       secondary_rc;           /* secondary return code          */
   unsigned char   ls_name[8];             /* name of link station           */
   LS_DEF_DATA     def_data;               /* LS defined data                */
} DEFINE_LS;
```

```
typedef struct ls_def_data
{
   unsigned char   description[32];        /* resource description           */
   unsigned char   initially_active;       /* is this LS initially active?   */
   unsigned char   reserv2;                /* reserved                       */
   AP_UINT16       react_timer;            /* timer for retrying failed LS   */
   AP_UINT16       react_timer_retry;      /* retry count for failed LS      */
   AP_UINT16       activation_count;       /* reserved                       */
   unsigned char   restart_on_normal_deact;  /* restart the link on any      */
                                             /* failure                      */
   unsigned char   reserv3[7];             /* reserved                       */
   unsigned char   port_name[8];           /* name of associated port        */
   unsigned char   adj_cp_name[17];        /* adjacent CP name               */
   unsigned char   adj_cp_type;            /* adjacent node type             */
   LINK_ADDRESS    dest_address;           /* destination address            */
   unsigned char   auto_act_supp;          /* auto-activate supported        */
   unsigned char   tg_number;              /* pre-assigned TG number         */
   unsigned char   limited_resource;       /* limited resource               */
   unsigned char   solicit_sscp_sessions;  /* solicit SSCP sessions          */
   unsigned char   pu_name[8];             /* Local PU name (reserved if     */
                                           /*  solicit_sscp_sessions is set  */
                                           /*  to AP_NO)                     */
   unsigned char   disable_remote_act;     /* disable remote activation      */
   unsigned char   dspu_services;          /* Services provided for          */
                                           /*  downstream PU                 */
   unsigned char   dspu_name[8];           /* Downstream PU name (reserved   */
                                           /*  if dspu_services is AP_NONE)  */
   unsigned char   dlus_name[17];          /* DLUS name if dspu_services     */
```

```
                                        /*  set to AP_DLUR               */
   unsigned char   bkup_dlus_name[17];  /* Backup DLUS name if           */
                                        /*  dspu_services set to AP_DLUR */
   unsigned char   hpr_supported;       /* does the link support HPR?    */
   unsigned char   hpr_link_lvl_error;  /* does the link use link-level  */
                                        /*  error recovery for HPR frms? */
   AP_UINT16       link_deact_timer;    /* link deactivation timer       */
   unsigned char   reserv1;             /* reserved                      */
   unsigned char   default_nn_server;   /* default LS to NN server?      */
   unsigned char   ls_attributes[4];    /* LS attributes                 */
   unsigned char   adj_node_id[4];      /* adjacent node ID              */
   unsigned char   local_node_id[4];    /* local node ID                 */
   unsigned char   cp_cp_sess_support;  /* CP-CP session support         */
   unsigned char   use_default_tg_chars;/* Use the default tg_chars      */
   TG_DEFINED_CHARS tg_chars;           /* TG characteristics            */
   AP_UINT16       target_pacing_count; /* target pacing count           */
   AP_UINT16       max_send_btu_size;   /* maximum send BTU size         */
   unsigned char   ls_role;             /* link station role             */
   unsigned char   max_ifrm_rcvd;       /* no. before acknowldgement     */
   AP_UINT16       dlus_retry_timeout;  /* seconds to recontact a DLUS   */
   AP_UINT16       dlus_retry_limit;    /* attempts to recontact a DLUS  */
   unsigned char   conventional_lu_compression; /* compression for LU 0-3? */
   unsigned char   conventional_lu_cryptography; /* reserved             */
   unsigned char   reserv3a;              /* reserved                    */
   unsigned char   retry_flags;           /* reserved                    */
   AP_UINT16       max_activation_attempts;  /* reserved                 */
   AP_UINT16       activation_delay_timer;   /* reserved                 */
   unsigned char   branch_link_type;      /* is link an up or down link  */
   unsigned char   adj_brnn_cp_support;   /* adj CP allowed to be BrNN?  */
   unsigned char   mltg_pacing_algorithm; /* reserved                    */
   unsigned char   reserv5;               /* reserved                    */
   AP_UINT16       max_rcv_btu_size;      /* reserved                    */
   unsigned char   tg_sharing_prohibited; /* reserved                    */
   unsigned char   link_spec_data_format; /* reserved                    */
   unsigned char   pu_can_send_dddlu_offline;  /* does the PU send NMVT  */
                                          /* (power off) to host?        */
   unsigned char   reserv4[13];           /* reserved                    */
   AP_UINT16       link_spec_data_len;    /* length of link specific data */
} LS_DEF_DATA;
```

```
typedef struct tg_defined_chars
{
   unsigned char   effect_cap;          /* effective capacity            */
   unsigned char   reserve1[5];         /* reserved                      */
   unsigned char   connect_cost;        /* connection cost               */
   unsigned char   byte_cost;           /* byte cost                     */
   unsigned char   reserve2;            /* reserved                      */
   unsigned char   security;            /* security                      */
   unsigned char   prop_delay;          /* propagation delay             */
   unsigned char   modem_class;         /* reserved                      */
   unsigned char   user_def_parm_1;     /* user-defined parameter 1      */
   unsigned char   user_def_parm_2;     /* user-defined parameter 2      */
   unsigned char   user_def_parm_3;     /* user-defined parameter 3      */
} TG_DEFINED_CHARS;
```

```
typedef struct link_address
{
  unsigned char   format;               /* type of link address         */
  unsigned char   reserve1;             /* reserved                     */
  AP_UINT16       length;               /* length                       */
  unsigned char   address[135];         /* address                      */
} LINK_ADDRESS;
```

DLC-specific data for SDLC:

```
typedef struct sdl_link_spec_data
{
   V0_MUX_INFO     mux_info;            /* Streams config info          */
   AP_UINT16       reserve8;            /* reserved                     */
   AP_UINT16       reserve9;            /* reserved                     */
   AP_UINT32       contact_timer;       /* reserved                     */
   AP_UINT16       contact_timer_retry; /* reserved                     */
   AP_UINT16       reserve1;            /* reserved                     */
   AP_UINT32       contact_timer2;      /* reserved                     */
   AP_UINT16       contact_timer_retry2;/* reserved                     */
   AP_UINT16       reserve2;            /* reserved                     */
   AP_UINT32       disc_timer;          /* reserved                     */
   AP_UINT16       disc_timer_retry;    /* reserved                     */
```

```
   AP_UINT16       reserve3;               /* reserved                       */
   AP_UINT32       nve_poll_timer;         /* reserved                       */
   AP_UINT16       nve_poll_timer_retry;   /* reserved                       */
   AP_UINT16       reserve4;               /* reserved                       */
   AP_UINT32       nve_poll_timer2;        /* reserved                       */
   AP_UINT16       nve_poll_timer_retry2;  /* reserved                       */
   AP_UINT16       reserve5;               /* reserved                       */
   AP_UINT32       no_resp_timer;          /* reserved                       */
   AP_UINT16       no_resp_timer_retry;    /* reserved                       */
   AP_UINT16       reserve6;               /* reserved                       */
   AP_UINT32       rem_busy_timer;         /* reserved                       */
   AP_UINT16       rem_busy_timer_retry;   /* reserved                       */
   unsigned char   re_tx_threshold;        /* reserved                       */
   unsigned char   repoll_threshold;       /* reserved                       */
   AP_UINT32       rr_timer;               /* reserved                       */
   unsigned char   group_address;          /* reserved                       */
   unsigned char   poll_frame;             /* Poll frame to use when Primary */
                                           /*  and contact polling secondary */
                                           /*  XID, SNRM                     */
   AP_UINT16       poll_on_iframe;         /* reserved                       */
   AP_UINT16       stub_spec_data_len;     /* reserved                       */
   STUB_SPEC_DATA  stub_spec_data;         /* reserved                       */
} SDL_LINK_SPEC_DATA;
```

DLC-specific data for QLLC:

```
typedef struct vql_ls_spec_data
{
   V0_MUX_INFO    mux_info;                /* streams config info            */
   AP_UINT16      reserve1;                /* reserved                       */
   AP_UINT16      reserve2;                /* reserved                       */
   unsigned char  vc_type;                 /* Virtual Circuit type           */
   unsigned char  req_rev_charge;          /* reserved                       */
   unsigned char  loc_packet;              /* reserved                       */
   unsigned char  rem_packet;              /* reserved                       */
   unsigned char  loc_wsize;               /* reserved                       */
   unsigned char  rem_wsize;               /* reserved                       */
   AP_UINT16      fac_len;                 /* X.25 facilities length         */
   unsigned char  fac[128];                /* X.25 facilities                */
   AP_UINT16      retry_limit;             /* reserved                       */
   AP_UINT16      retry_timeout;           /* reserved                       */
   AP_UINT16      idle_timeout;            /* reserved                       */
   AP_UINT16      pvc_id;                  /* PVC logical channel identifier */
   AP_UINT16      sn_id_len;               /* reserved                       */
   unsigned char  sn_id[4];                /* reserved                       */
   AP_UINT16      cud_len;                 /* length of any call user data   */
                                           /*  to send                       */
   unsigned char  cud[128];                /* actual call user data          */
   AP_UINT32      xtras;                   /* reserved                       */
   AP_UINT32      xtra_len;                /* reserved                       */
   unsigned char  rx_thruput_class;        /* reserved                       */
   unsigned char  tx_thruput_class;        /* reserved                       */
   unsigned char  cugo;                    /* reserved                       */
   unsigned char  cug;                     /* reserved                       */
   AP_UINT16      cug_index;               /* reserved                       */
   AP_UINT16      nuid_length;             /* reserved                       */
   unsigned char  nuid_data[109];          /* reserved                       */
   unsigned char  reserve3[2];             /* reserved                       */
   unsigned char  rpoa_count;              /* reserved                       */
   AP_UINT16      rpoa_ids[30];            /* reserved                       */
} VQL_LS_SPEC_DATA;
```

DLC-specific data for Token Ring, Ethernet:

```
typedef struct llc_link_spec_data
{
   V0_MUX_INFO     mux_info;               /* Streams config info            */
   AP_UINT16       reserve1;               /* reserved                       */
   AP_UINT16       reserve2;               /* reserved                       */
   AP_UINT16       length;                 /* reserved                       */
   AP_UINT16       xid_timer;              /* XID timeout value in seconds   */
   AP_UINT16       xid_timer_retry;        /* XID retry limit                */
   AP_UINT16       test_timer;             /* TEST timeout value in seconds  */
   AP_UINT16       test_timer_retry;       /* TEST retry limit               */
   AP_UINT16       ack_timeout;            /* acknowledgment timeout in ms   */
   AP_UINT16       p_bit_timeout;          /* POLL response timeout in ms    */
   AP_UINT16       t2_timeout;             /* acknowledgment delay in ms     */
   AP_UINT16       rej_timeout;            /* REJ response timeout in seconds*/
   AP_UINT16       busy_state_timeout;     /* remote busy timeout in seconds */
```

```
   AP_UINT16       idle_timeout;           /* idle RR interval in seconds    */
   AP_UINT16       max_retry;              /* retry limit for any response   */
} LLC_LINK_SPEC_DATA;
```

DLC-specific data for multipath channel (MPC), CS Linux for IBM Z only:

```
typedef struct chnl_link_spec_data
{
   V0_MUX_INFO     mux_info;               /* streams information            */
   AP_UINT16       device_end;             /* BlkMux protocol flag           */
   unsigned char   escd_port;              /* reserved                       */
   unsigned char   cuadd;                  /* reserved                       */
   unsigned char   local_name[8];          /* reserved                       */
   unsigned char   remote_name[8];         /* reserved                       */
   unsigned char   reserv1[32];            /* pad and future expansion       */
} CHNL_LINK_SPEC_DATA;
```

DLC-specific data for Enterprise Extender (HPR/IP):

```
typedef struct ipdlc_link_spec_data
{
   V0_MUX_INFO     mux_info;              /* streams information             */
   AP_UINT16       ack_timeout;           /* ACK timer for command frames    */
   AP_UINT16       max_retry;             /* Retry limit for command frames  */
   AP_UINT16       liveness_timeout;      /* Liveness timer                  */
   unsigned char   short_hold_mode;       /* Run in short-hold mode          */
   unsigned char   remote_hostname[255];  /* Name of remote host to contact  */
} IPDLC_LINK_SPEC_DATA;
```

Data for all DLC types:

```
typedef struct v0_mux_info
{
   AP_UINT16         dlc_type;               /* DLC implementation type     */
   unsigned char     need_vrfy_fixup;        /* reserved                    */
   unsigned char     num_mux_ids;            /* reserved                    */
   AP_UINT32         card_type;              /* type of adapter card        */
   AP_UINT32         adapter_number;         /* DLC adapter number          */
   AP_UINT32         oem_data_length;        /* reserved                    */
   AP_INT32          mux_ids[5];             /* reserved                    */
} V0_MUX_INFO;
```

For Token Ring or Ethernet, the *address* parameter in the `link_address` structure is replaced by the following:

```
typedef struct tr_address
{
   unsigned char   mac_address[6];         /* MAC address                    */
   unsigned char   lsap_address;           /* local SAP address              */
} TR_ADDRESS;
```

For Enterprise Extender (HPR/IP), the *address* parameter in the `link_address` structure is replaced by the following:

```
typedef struct ip_address_info
{
  unsigned char lsap;                      /* Local Service Access Point addr */
  unsigned char version;                   /* IPv4 or IPv6                    */
  unsigned char address[272];              /* reserved                        */

} IP_ADDRESS_INFO;
```

For MPC, the `link_address` structure is reserved.

Supplied parameters

The application supplies the following parameters:

opcode
AP_DEFINE_LS

ls_name
Name of link station. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes.

def_data.description
A null-terminated text string (0-31 characters followed by a null character) describing the LS. This string is for information only; it is stored in the node's configuration file and returned on the QUERY_LS, QUERY_PU, and QUERY_DOWNSTREAM_PU verbs, but CS Linux does not make any other use of it.

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

AP_YES
The LS is automatically started when the node is started.

AP_NO
The LS is not automatically started; it must be started manually.

If the LS is a leased SDLC link or a QLLC PVC link, you are recommended to set this parameter to AP_YES to ensure that the link is always available.

def_data.react_timer
Reactivation timer for reactivating a failed LS. If the *react_timer_retry* parameter below is nonzero, to specify that CS 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, CS Linux waits for the specified time before retrying the activation. If *react_timer_retry* is zero, this parameter is ignored.

def_data.react_timer_retry
Retry count for reactivating a failed LS. This parameter is used to specify whether CS Linux should attempt to reactivate the LS if it fails while in use (or if an attempt to start the LS fails).

Specify zero to indicate that CS Linux should not attempt to reactivate the LS, or specify the number of retries to be made. A value of 65,535 indicates that CS Linux should retry indefinitely until the LS is activated.

CS Linux waits for the time specified by the *react_timer* parameter above between successive retries. If the retry count is reached without successfully reactivating the LS, or if a STOP_LS is issued while CS 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 AP_YES, the reactivation timer fields are ignored; if the link fails, CS 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 or a QLLC PVC link, you are recommended to set this parameter to a non-zero value to ensure that the link is always available.

def_data.restart_on_normal_deact
Specifies whether CS Linux should attempt to reactivate the LS if it is deactivated normally by the remote system. Possible values are:

AP_YES
If the remote system deactivates the LS normally, CS 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).

AP_NO
If the remote system deactivates the LS normally, CS Linux does not attempt to reactivate it.

If the LS is a host link (specified by the *def_data.adj_cp_type* parameter), or is automatically started when the node is started (the *initially_active* parameter is set to AP_YES), this parameter is ignored; CS Linux always attempts to reactivate the LS if it is deactivated normally by the remote system (unless *react_timer_retry* is zero).

def_data.port_name
Name of port associated with this link station. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes, which must match the name of a defined port.

def_data.adj_cp_name
Fully qualified name of the adjacent CP for this LS.

If the *adj_cp_type* parameter below is set to AP_NETWORK_NODE or AP_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 AP_BACK_LEVEL_LEN_NODE, CS Linux uses this value only as an identifier; set it to any string (of the format described below) 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; CS Linux will check the CP name only if one is specified.

The name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of 1-8 A-string characters, an EBCDIC dot (period) character, and a network name of 1-8 A-string characters.

def_data.adj_cp_type
Adjacent node type.

If the adjacent node is an APPN node, and preassigned TG numbers are not being used, this is normally set to AP_APPN_NODE, indicating that the node type is unknown; CS 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, CS Linux will reject a connection attempt from the adjacent node if its node type does not match the one specified here. Use one of the following values:

AP_APPN_NODE
The node type is unknown. CS Linux will determine the type during XID exchange.

AP_END_NODE
End node, Branch Network Node acting as an End Node from the local node's perspective, or up-level LEN node (one that includes the Network Name CV in its XID3).

AP_NETWORK_NODE
Network node, or 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. These values are not valid for an Enterprise Extender, MPC link, which must be to an APPN node.

AP_BACK_LEVEL_LEN_NODE
Back-level LEN node (one that does not include the Network Name CV in its XID3).

AP_HOST_XID3
Host node; CS Linux should respond to a polling XID from the node with a format 3 XID.

AP_HOST_XID0
Host node; CS Linux should respond to a polling XID from the node with a format 0 XID.

AP_DSPU_XID
Downstream PU; CS Linux should include XID exchange in link activation. The *dspu_name* and *dspu_services* fields must also be set.

AP_DSPU_NOXID
Downstream PU; CS Linux should not include XID exchange in link activation. The *dspu_name* and *dspu_services* fields 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 AP_APPN_NODE, AP_END_NODE, AP_NETWORK_NODE, or AP_BACK_LEVEL_LEN_NODE.

def_data.dest_address.format
For MPC, this parameter is reserved.

The type of link address specified. Possible values:

AP_IP_ADDRESS_INFO
IP address. Specify this value for an Enterprise Extender (HPR/IP) link.

AP_UNSPECIFIED
Unspecified address format. Specify this value for any link type other than Enterprise Extender (HPR/IP).

def_data.dest_address.length
For MPC, this parameter is reserved.

Length of the destination address field, as specified in the following parameter or parameters.

For Enterprise Extender (HPR/IP), this parameter and *dest_address.address* are reserved. Instead, you specify the address using the *remote_hostname* parameter in the link-specific data.

For SDLC:

def_data.dest_address.address
Address of the secondary station on this LS.

- If the port that owns this LS is used only for incoming calls (*out_link_act_lim* on DEFINE_PORT is zero), this parameter is reserved.
- If the port that owns this LS is switched primary and is used for outgoing calls (*port_type* is AP_SWITCHED, *ls_role* is AP_LS_PRI, and *out_link_act_lim* on DEFINE_PORT is nonzero), 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 which must match the value configured at the secondary station.
- Otherwise, set it 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_PORT is AP_LS_PRI and *tot_link_act_lim* is greater than 1), this address must be different for each LS on the port.

For QLLC:

def_data.dest_address.address
Address of the destination node for this LS. This parameter is used only for SVC outgoing calls (defined by the *vc_type* parameter in the link-specific data, and by the link activation limit parameters on DEFINE_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.

For Token Ring, Ethernet:

def_data.dest_address.mac_address
MAC address of 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), set this parameter to a null string. 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.

If the local and adjacent nodes are on LANs of different types (one Token Ring , the other Ethernet) connected by a bridge, you will probably need to reverse the bit order of the bytes in the MAC address. For more information, see “Bit ordering in MAC addresses” on page 124. If the two nodes are on the same LAN, or on LANs of the same type connected by a bridge, no change is required.

def_data.dest_address.lsap_address
Local SAP address of adjacent node. This must be a multiple of 0x04.

For Enterprise Extender (HPR/IP):

def_data.dest_address.ip_address_info.lsap
For Enterprise Extender: Local SAP address of the port. Specify a multiple of 0x04 in the range 0x04-0xEC. The usual value is 0x04, but VTAM may use 0x08 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.

def_data.dest_address.ip_address_info.version
For Enterprise Extender: Specifies whether the following field represents an IPv4 or IPv6 address. Possible values:

IP_VERSION_4_HOSTNAME
The *address* field specifies an IPv4 address, or a hostname or alias that resolves to an IPv4 address.

IP_VERSION_6_HOSTNAME
The *address* field specifies an IPv6 address, or a hostname or alias that resolves to an IPv6 address.

For all link types:

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

AP_YES
The link can be activated automatically.

AP_NO
The link cannot be activated automatically.

If this parameter is set to AP_YES:

- The reactivation timer fields are ignored. If the LS fails, CS 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 will not be reactivated by CS Linux, and must be restarted manually.
- If the link is to an APPN node, the LS must have a preassigned TG number defined (see the following parameter), and *cp_cp_sess_support* must be set to AP_NO.
- If either the local node or the adjacent node is an end node, the LS must also be defined as auto-activatable at the adjacent node.

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

This TG number is 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.

If the local node is a LEN node, or if the adjacent node is a LEN node and the link is to be auto-activatable, set the TG number to 1. Otherwise, specify a number in the range 1-20, or 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 AP_END_NODE or AP_NETWORK_NODE.

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

AP_NO
The link is not a limited resource and will not be deactivated automatically.

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

AP_INACTIVITY
The link is a limited resource and will be deactivated automatically 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* field.

- 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* field of the DEFINE_NODE verb.

A limited resource link station may be configured for CP-CP session support, by setting this field to AP_NO_SESSIONS and *cp_cp_sess_support* to AP_YES. In this case, if CP-CP sessions are brought up over the link, CS Linux will not treat the link as a limited resource (and so will not deactivate it).

def_data.solicit_sscp_sessions
For an Enterprise Extender (HPR/IP) or MPC port, this parameter is reserved.

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 AP_NETWORK_NODE or AP_END_NODE); it is ignored otherwise. If the adjacent node is a host (*adj_cp_type* is either AP_HOST_XID3 or AP_HOST_XID0), CS Linux always requests the host to initiate SSCP sessions.

Possible values are:

AP_YES
Request the adjacent node to initiate SSCP sessions.

AP_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 AP_NONE, this parameter must be set to AP_NO.

def_data.pu_name
For an Enterprise Extender (HPR/IP) or MPC port, this parameter is reserved.

Name of the local PU that uses this link. This parameter is used only if *adj_cp_type* is set to AP_HOST_XID3 or AP_HOST_XID0, or if *solicit_sscp_sessions* is set to AP_YES; it is ignored otherwise. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces.

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

AP_YES
The LS can only be activated by the local node; if the remote node attempts to activate it, CS Linux will reject the attempt.

AP_NO
The LS can be activated by the remote node.

def_data.dspu_services
For an Enterprise Extender (HPR/IP) or MPC port, this parameter is reserved.

Specifies the services which the local node will provide 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 AP_NO; it is reserved otherwise. Possible values are:

AP_PU_CONCENTRATION
Local node will provide SNA gateway for the downstream PU. The local node must be defined to support SNA gateway.

AP_DLUR
Local node will provide DLUR services for the downstream PU. The local node must be defined to support DLUR. (Not supported on end node.)

AP_NONE
Local node will provide no services for this downstream PU.

def_data.dspu_name
For an Enterprise Extender (HPR/IP) or MPC port, this parameter is reserved.

Name of the downstream PU. The name is an 8-byte type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

This parameter is required when both of the following conditions are true; otherwise, it is reserved:

- The *solicit_sscp_sessions* parameter is set to AP_NO
- The *dspu_services* parameter is set to AP_PU_CONCENTRATION or AP_DLUR

If the downstream PU is used for DLUR, this name should match the PU name configured on the host. (CS 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.)

def_data.dlus_name
For an Enterprise Extender (HPR/IP) or MPC port, this parameter is reserved.

Name of the DLUS node from which DLUR solicits SSCP services when the link to the downstream node is activated. This field is reserved if *dspu_services* is not set to AP_DLUR.

The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

To specify the global default DLUS, defined using the DEFINE_DLUR_DEFAULTS verb, set this parameter to 17 binary zeros. If this parameter is set to zeros and there is no global default DLUS, then DLUR will not initiate SSCP contact when the link is activated.

def_data.bkup_dlus_name
For an Enterprise Extender (HPR/IP) or MPC port, this parameter is reserved.

Name of the backup DLUS node from which DLUR solicits SSCP services if the node specified by *dlus_name* is not active. This field is reserved if *dspu_services* is not set to AP_DLUR.

The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

To specify the global backup default DLUS, defined using the DEFINE_DLUR_DEFAULTS verb, set this parameter to 17 binary zeros.

def_data.hpr_supported
Specifies whether HPR is supported on this link. If the link is an Enterprise Extender (HPR/IP) link, this parameter must be set to AP_YES. If it is an MPC link, this parameter must be set to AP_NO. Otherwise, it must be set to AP_NO unless the *adj_cp_type* parameter indicates that the link connects to an APPN node. Possible values are:

AP_YES
HPR is supported on this link.

AP_NO
HPR is not supported on this link.

def_data.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 AP_YES.

This parameter is reserved for SDLC / Channel/ MPC+ / Enterprise Extender (HPR/IP) links.

Possible values are:

AP_YES
HPR traffic should be sent on this link using link-level error recovery.

AP_NO
HPR traffic should not be sent on this link using link-level error recovery.

def_data.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 AP_YES, the default deactivation timer value of 30 is used.
- If the *hpr_supported* parameter is set to AP_NO, no timeout is used (the link is not deactivated, as if *limited_resource* were set to AP_NO).

def_data.default_nn_server
End node: 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 LSs to find a suitable LS to activate. This allows 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:

AP_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 AP_YES.

AP_NO
This link should not be automatically activated in an attempt to contact a network node server.

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

def_data.ls_attributes
This array contains further information about the adjacent node, as described in the following parameters:

def_data.ls_attributes[0]
Host type. Set this to AP_SNA unless you are communicating with a host of one of the other types listed below. Possible values are:

AP_SNA
Standard SNA host.

AP_FNA
Fujitsu Network Architecture (VTAM-F) host.

AP_HNA
Hitachi Network Architecture host.

def_data.ls_attributes[1]
Network Name CV suppression for a link to a back-level LEN node.

If *adj_cp_type* is set to AP_BACK_LEVEL_LEN_NODE or AP_HOST_XID3, specify whether to suppress inclusion of the Network Name CV in the format 3 XID sent to the LEN node, using one of the following values:

AP_NO
Include the Network Name CV in the XID.

AP_SUPPRESS_CP_NAME
Do not include the Network Name CV.

If *adj_cp_type* is set to any other value, this parameter is ignored.

def_data.adj_node_id
For an MPC link, this parameter is reserved.

Node ID of adjacent node. This is a 4-byte hexadecimal string, consisting of a block number (three hexadecimal digits) and a node number (five hexadecimal digits). Set it to zeros to disable node ID checking. If this link station is defined on a switched port, the *node_id* must be unique, and there may only be one null *node_id* on each switched port.

def_data.local_node_id
For an MPC link, this parameter is reserved.

Node ID sent in XIDs on this LS. This is a 4-byte hexadecimal string, consisting of a block number (3 hexadecimal digits) and a node number (5 hexadecimal digits). Set it to zeros to use the node ID specified in the DEFINE_NODE verb.

def_data.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 AP_NETWORK_NODE, AP_END_NODE, or AP_APPN_NODE); it is ignored otherwise. If both the local node and the adjacent node are network nodes, this parameter should be set to AP_YES in order to use APPN functions between these nodes.

Possible values are:

AP_YES
CP-CP sessions are supported. For an MPC or MPC+ LS, the *solicit_sscp_sessions* parameter must be set to AP_NO.

AP_NO
CP-CP sessions are not supported.

def_data.use_default_tg_chars
Specifies whether the default TG characteristics supplied on the DEFINE_PORT verb should be used. The TG characteristics apply only if the link is to an APPN node; this parameter, and the parameters in the tg_chars structure, are ignored otherwise. Possible values are:

AP_YES
Use the default TG characteristics; ignore the tg_chars structure on this verb.

AP_NO
Use the tg_chars structure on this verb.

def_data.tg_chars.effect_cap
Actual bits per second rate (line speed). The value is encoded as a 1-byte floating point number, represented by the formula 0.1 mmm * 2 ^{eeeee} where the bit representation of the byte is b'eeeeemmm'. Each unit of effective capacity is equal to 300 bits per second.

For an Ethernet or Enterprise Extender (HPR/IP) 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.

def_data.tg_chars.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.

def_data.tg_chars.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.

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

AP_SEC_NONSECURE
No security.

AP_SEC_PUBLIC_SWITCHED_NETWORK
Data is transmitted over a public switched network.

AP_SEC_UNDERGROUND_CABLE
Data is transmitted over secure underground cable.

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

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

AP_SEC_ENCRYPTED
Data is encrypted before transmission over the line.

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

def_data.tg_chars.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:

AP_PROP_DELAY_MINIMUM
Minimum propagation delay.

AP_PROP_DELAY_LAN
Delay is less than 480 microseconds (typical for a LAN).

AP_PROP_DELAY_TELEPHONE
Delay is in the range 480-49,512 microseconds (typical for a telephone network).

AP_PROP_DELAY_PKT_SWITCHED_NET
Delay is in the range 49,512-245,760 microseconds (typical for a packet-switched network).

AP_PROP_DELAY_SATELLITE
Delay is greater than 245,760 microseconds (typical for a satellite link).

AP_PROP_DELAY_MAXIMUM
Maximum propagation delay.

def_data.tg_chars.user_def_parm_1 **through** ***def_data.tg_chars.user_def_parm_3***
User-defined parameters, which you can use to include other TG characteristics not covered by the above parameters. Each of these parameters must be set to a value in the range 1-255.

def_data.target_pacing_count
Numeric value between 1 and 32,767 inclusive indicating the desired pacing window size. (The current version of CS Linux does not make use of this value.)

def_data.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 (265-4105 for SDLC).

def_data.ls_role
Link station role. This is normally set to AP_USE_PORT_DEFAULTS, specifying that the LS role is to be taken from the definition of the port that owns this LS. For an MPC link, this is the only valid value.

If you need to override the port's LS role for an individual LS, specify one of the following values:

AP_LS_PRI
Primary

AP_LS_SEC
Secondary

AP_LS_NEG
Negotiable

For an Enterprise Extender (HPR/IP) port, you must use AP_USE_PORT_DEFAULTS; you cannot override the port's LS role.

def_data.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 the port definition is used.

def_data.dlus_retry_timeout
For an Enterprise Extender (HPR/IP) or MPC port, this parameter is reserved.

Reactivation timer for contacting a DLUS. If CS Linux fails to contact the DLUS, this parameter specifies the time in seconds between retries.

Specify a value in the range 0x0001-0xFFFF.

def_data.dlus_retry_limit
For an Enterprise Extender (HPR/IP) or MPC port, this parameter is reserved.

The interval in seconds between the second and subsequent attempts to contact the DLUS specified by the *dlus_name* and *bkup_dlus_name* parameters. Specify a value in the range 0x0001-0xFFFE, or specify 0xFFFF to indicate that CS Linux should retry indefinitely until it contacts the DLUS. The interval between the first and second attempts is always 1 second. If zero is specified, then the defaults specified using the DEFINE_DLUR_DEFAULTS verb are used. This parameter is ignored if the *dspu_services* parameter is not set to AP_DLUR.

def_data.conventional_lu_compression
For an MPC link, this parameter is reserved.

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:

AP_YES
Data compression should be used for LU 0-3 sessions on this link if the host requests it.

AP_NO
Data compression should not be used for LU 0-3 sessions on this link.

def_data.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 *def_data.adj_cp_type* is set to AP_NETWORK_NODE, AP_END_NODE, AP_APPN_NODE, or AP_BACK_LEVEL_LEN_NODE, this parameter defines whether the link is an uplink or a downlink. Possible values are:

AP_UPLINK
The link is an uplink.

AP_DOWNLINK
The link is a downlink.

If *def_data.adj_cp_type* is set to AP_NETWORK_NODE, this parameter must be set to AP_UPLINK.

def_data.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 *def_data.adj_cp_type* is set to AP_NETWORK_NODE, or it is set to

AP_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:

AP_BRNN_ALLOWED
The adjacent node is allowed (but not required) to be a Branch Network Node.

AP_BRNN_REQUIRED
The adjacent node must be a Branch Network Node.

AP_BRNN_PROHIBITED
The adjacent node must not be a Branch Network Node.

If *def_data.adj_cp_type* is set to AP_NETWORK_NODE and *auto_act_supp* is set to AP_YES, this parameter must be set to AP_BRNN_REQUIRED or AP_BRNN_PROHIBITED.

def_data.pu_can_send_dddlu_offline
For an MPC link, this parameter is reserved.

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), CS 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 AP_YES and *dspu_services* is not set to AP_NONE).

Possible values are:

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

AP_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 AP_NO.

def_data.link_spec_data_len
Length of the link-specific data. The data should be concatenated to the basic structure.

Link-specific data for SDLC:

mux_info.dlc_type
Type of the DLC. Set this to AP_IMPL_SDLC_SL.

poll_frame
The frame to use for pre-activation polling. This is normally XID, indicating that polling is in the control of the DLC user. However, when CS Linux is primary talking to an old secondary implementation, it may be necessary to poll using some other frame. Possible values are: XID, SNRM.

Link-specific data for QLLC:

mux_info.dlc_type
Type of the DLC. Set this to AP_IMPL_NLI_QLLC.

vc_type
The Virtual Circuit type of the LS. Possible values are:

VQL_SVC
Switched Virtual Circuit

VQL_PVC
Permanent Virtual Circuit

If you define both SVC and PVC LSs between the same local node and remote node, unpredictable results may occur if the SVC LS is started first (since it may not be possible to match the incoming call

to the correct LS). To avoid these problems, ensure that PVC LSs are activated before any SVC LSs between the same pair of nodes.

fac_len
Length of the additional X.25 facilities data that follows (in the *fac* parameter). If no additional data is required, specify zero. If the X.25 network does not support facilities negotiation, specify zero and see the *fac* parameter below for more information.

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 to a decimal number to identify which PVC (from the range of PVCs defined for your X.25 provider software) is to be used for this LS. This field is reserved if *vc_type* above is set to VQL_SVC.

cud_len
Length of the Call User Data that follows (in the *cud* parameter).

cud
Call User Data: this parameter identifies the protocol to be used over the underlying X.25 virtual circuit, and is used only if the *vc_type* parameter is set to VQL_SVC. For most implementations, this should be 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; check with the System Administrator of the remote system.

DLC-specific data for Token Ring, Ethernet:

mux_info.dlc_type
Type of the DLC.

Possible values are:

AP_IMPL_TR_SNAP_LLC2
Token Ring

AP_IMPL_ETHER_SNAP_LLC2
Ethernet

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 remote station is on a separate Token Ring connected by a bridge.

xid_timer_retry
Number of times transmission and retransmission of an XID is allowed. This count does not include the initial transmission; that is, a value of 1 indicates "transmit once and then retry once". Higher values may be needed if the remote station is on a separate Token Ring connected by a bridge.

test_timer
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 remote station is on a separate Token Ring connected by a bridge.

test_timer_retry
Number of times transmission and retransmission of a TEST frame is allowed. This count does not include the initial transmission; that is, a value of 1 indicates "transmit once and then retry once". Higher values may be needed if the remote station is on a separate Token Ring connected by a bridge.

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 allows the local station to respond to more than one I-frame with a single RR, and so reduces acknowledgment traffic.

rej_timeout
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 time in seconds that the local station waits for indication from the adjacent link station that a busy state (RNR) has cleared.

idle_timeout
Idle timeout: 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.

Link-specific data for multipath channel (MPC), CS Linux for IBM Z only:

chnl_link_spec_data.mux_info.dlc_type
Type of DLC. This must be set to AP_IMPL_MPC_GDLC.

Link-specific data for Enterprise Extender (HPR/IP):

ipdlc_link_spec_data.mux_info.dlc_type
Type of DLC. Set this to AP_IP.

ipdlc_link_spec_data.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 10000 milliseconds.

ipdlc_link_spec_data.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.

ipdlc_link_spec_data.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).

ipdlc_link_spec_data.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 (AP_YES or AP_NO).

Setting this parameter to AP_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 AP_NO.

ipdlc_link_spec_data.remote_hostname
Remote host name of the destination node for this link. 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 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).

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

追加の変更ポート名
ls_name パラメーターが既存の LS の名前と一致しましたが、 ポート名 パラメーターが既存の定義と一致しませんでした。 既存の LS の定義の変更時にポート名を変更することはできません。

リンク・ リンクの検証 / セキュリティーが無効です
*tg_chars.*セキュリティー パラメーターが有効な値に設定されていませんでした。

追加の AUTO_AUTO_ACT_SUPP
auto_act_supp パラメーターが有効な値に設定されていなかったか、または *cp_cp_sess_support* が **類人猿**に設定されているときに **類人猿** に設定されていました。

ファイルの検証 CP_NAME
付加属性名 パラメーターに、無効な文字が含まれていたか、正しい形式ではなかったか、または必要なときに指定されませんでした。

リソースの追加が無効
limited_resource パラメーターが有効な値に設定されていませんでした。

リンク名の入力
ls_name パラメーターに、無効な文字が含まれていました。

追加の追加のロール
ls_role パラメーターが有効な値に設定されていませんでした。

アプリケーション・ ノード・ タイプの追加
付加 *cp_type* パラメーターが有効な値に設定されていませんでした。

ポートフォリオ・ ポート名
ポート名 パラメーターが、定義されたどのポートの名前とも一致しませんでした。

エイプインバリデータプール名
プール名 パラメーターが定義された PU の名前と一致しなかったか、または既に定義されている LS 上の新しい値に設定されました。

ファイルの追加に使用さない
dspu_name パラメーターが定義された PU の名前と一致しなかったか、または既に定義されている LS 上の新しい値に設定されました。

付加的なサービス・サービス
dspu_services パラメーターが有効な値に設定されていないか、予期されていないときに設定されました。

通知に使用されないことがある
赤字の *sscp_sess* パラメーターが有効な値に設定されていませんでした。

ファイルの追加が無効 （ _CNT ）
ターゲット・パック・カウント パラメーターが有効な値に設定されていませんでした。

ファイル名の入力が無効です
dlus_name パラメーターに、無効な文字が含まれているか、または正しい形式ではありませんでした。

アプインバリデータ・プクル名
bkup_dlus_name パラメーターに、無効な文字が含まれているか、または正しい形式ではありませんでした。

無効な TG__ 数
指定された TG 番号は有効な範囲内にありませんでした。

ID 名の指定が欠落しています
TG 番号が定義されましたが、 CP 名が指定されませんでした。

一致する CP_TYPE
TG 番号が定義されましたが、 CP タイプが指定されませんでした。

付加された TG_NUMBER
リンクは自動活動化されるように定義されていますが、 TG 番号が指定されていません。

アプリケーションでサポートされているものを追加します
このノードは、同じ隣接ノードとの間で定義された複数の LS をサポートできません。

エントリー・エントリーの再試行の制限
dlus_retry_limit に指定された値が無効でした。

追加の再試行がタイムアウトになります
dlus_retry_timeout に指定された値が無効でした。

追加の追加のロール
ls_role パラメーターに指定された値が無効です。

ノード・タイプ FOR_HP_FOR_HPR
付加 *cp_type* パラメーターに指定されたノード・タイプは HPR をサポートしていません。

ファイルのサイズが無効です
最大 *send_btu_size* パラメーターに指定された値が有効ではありませんでした。

エージェントの追加の無効化
max_ifrm_rcvd パラメーターに指定された値が有効ではありませんでした。

認識されていないホスト （ _R ）
この値は、 Enterprise Extender (HPR/IP) リンクにのみ適用されます。 リモート・ホスト名 パラメーターに指定されたストリングを、有効な IP アドレスに解決できませんでした。

ファイルの追加バージョン
この値は、 Enterprise Extender (HPR/IP) リンクにのみ適用されます。 *ip_version* パラメーターに指定された値が、所有 IP ポートに指定された値と一致しませんでした。

ファイル・リンク・リンク・タイプの追加
ブランチ・リンク・タイプ パラメーターが有効な値に設定されていませんでした。

サポートされているファイルの追加
付加 *cpnn_cp_support* パラメーターが有効な値に設定されていませんでした。

追加のサポートが欠落しています
付加 *cpnn_cp_support* パラメーターが **許可される** AP_BRNN__に設定されました。この値は、隣接ノードがネットワーク・ノードで、 *auto_act_supp* が **類人猿**に設定されているため、無効です。

使用中のアップリンク
ブランチ・リンク・タイプ パラメーターが **アプ· アップリンク**に設定されましたが、ローカル・ノードと隣接ノードの間の既存の LS の定義で、それが下位リンクであることが指定されています。 ブランチ・リンク・タイプは、同じ 2 つのノード間のすべての LS で同じでなければなりません。

追加リンク解除リンク
ブランチ・リンク・タイプ パラメーターが AP_DOWNLINK に設定されましたが、ローカル・ノードと隣接ノードの間の既存 LS の定義は、それがアップリンクであることを指定しています。 ブランチ・リンク・タイプは、同じ 2 つのノード間のすべての LS で同じでなければなりません。

リンク· リンクの形式が無効です
予約済みパラメーターがゼロ以外の値に設定されました。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター· チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc
可能な値は次のとおりです

コピー· ユーザー名の追加
付加属性名 パラメーターで指定された CP 名へのリンクは、すでに定義されています。

宛先アドレスの追加 （ _R ）
アドレス パラメーターに指定された宛先アドレスへのリンクが既に定義されています。

LLC2 リンク・タイプの場合 : *mac_* アドレス パラメーターと *lsap_* アドレス パラメーターの組み合わせによって指定された宛先アドレスへのリンクは、すでに定義されています。

付加属性の追加ノード ID
従属ノード *ID* (隣接ノードのノード ID) は、すでに別のリンク・ステーションに定義されています。

リンク名の入力
ls_name パラメーターに指定されたリンク・ステーションの値が無効でした。

指定された LS_INVALID_NUM_LS_LS_特定
指定されたリンク・ステーションの数が正しくありませんでした。

ロケール CP_NAME
付加属性名 パラメーターに指定された名前は、ローカル CP 名と同一です。

アクティブ （ LS_LS）
ls_name パラメーターに指定されているリンク・ステーションは、現在アクティブです。

定義済み AP_PU_ALREADY_DEFINED
プール名 パラメーターで指定された PU はすでに定義されています。

定義済み AP_DSPU_ALREADY_DEFINED
dspu_name パラメーターで指定されたダウンストリーム PU はすでに定義されています。

AP_DSPU_SERVICES_NOT_SUPPORTED
付加プールの集中パラメーターまたは **アプドラー** が *dspu_services* パラメーターに指定されていますが、ノードはそれをサポートしていません。

重複と重複の数 （ __ ）
tg_number パラメーターに指定された TG 番号は、既に定義されています。

使用中の TG_NUMBER_NUMBER_IN_USE
tg_number パラメーターに指定された TG 番号は、既に別の LS によって使用されています。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **状態検査の追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

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. Normally, the local and remote nodes are on the same LAN, or on LANs of the same type connected by a bridge; in this case, they 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 normally need to reverse the bit order of each byte of the address when specifying a remote MAC address. To do this, take the following steps:

1. List the MAC address as six bytes, each byte represented by two hexadecimal digits.
2. List the MAC address as six bytes, each byte represented by two hexadecimal digits.
3. Convert each digit as shown below:

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

```
Example of Bit Ordering in a MAC Address
```

Original address	1A	2B	3C	4D	5E	6F
Swap digits	A1	B2	C3	D4	E5	F6
Convert digits (the bit-reversed form of the original address)	58	D4	3C	B2	7A	F6

Modem control characters

For SDLC, if you need to include one or more non-printable control characters in the *hmod_data* parameter, you can do this by specifying the hexadecimal value of the control character, as listed in Table 2 on page 124.

Table 2. Escape Sequences for Modem Control Characters

Escape Sequence	Decimal Value	Hexadecimal Value
NUL	0	0x00
SOH	1	0x01

Table 2. Escape Sequences for Modem Control Characters (continued)

Escape Sequence	Decimal Value	Hexadecimal Value
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
DEL	127	0x7F

定義済み LS_ルーティング

DEFINE_LS_ROUTING verb は、リンク・ステーションを使用してパートナー LU のロケーションを定義します。

注 : ユーザーは、 Enterprise Extender (HPR/IP) リンク・ステーションで DEFINE_LS_ROUTING を使用することはできません。 これは、このリンク・タイプのすべてのトラフィックが RTP 接続を経由して流れている必要があるためです。 RTP 接続は、特定のリンク・ステーションに固定されておらず、別のパスに切り替えることができます。

VCB 構造体

```
typedef struct define_ls_routing
{
  AP_UINT16         opcode;               /* verb operation code           */
  unsigned char     reserv2;              /* reserved                      */
  unsigned char     format;               /* reserved                      */
  AP_UINT16         primary_rc;           /* primary return code           */
  AP_UINT32         secondary_rc;         /* secondary return code         */
  unsigned char     lu_name[8];           /* LU Name                       */
  unsigned char     lu_alias[8];          /* reserved                      */
  unsigned char     fq_partner_lu[17];    /* partner lu name               */
  unsigned char     wildcard_fqplu;       /* wildcard partner LU flag      */
  unsigned char     ls_name[8];           /* link to use                   */
  unsigned char     reserv3[2];           /* reserved                      */
} DEFINE_LS_ROUTING;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
追加の保留中のルーティング

lu_name
ls_name パラメーターで指定されたリンクを介して、パートナー LU (*fq_partner_lu* パラメーターで指定される) と通信するローカル LU の名前。 これは、 8 バイトのタイプ A の文字ストリングです。

fq_partner_lu
ローカル LU (*lu_name* パラメーターで指定される) が *ls_name* パラメーターで指定されたリンクを介して通信するパートナー LU の完全修飾名。 1 から 8 文字のネットワーク名、ピリオド、 1 から 8 文字のパートナー LU 名で構成される 3 文字から 17 文字を指定します。

名前の一部のみを指定し、 ワイルドカード・ディスク パラメーターを **類人猿**に設定することによって、ワイルドカード・パートナー LU 名の一部または全部を指定することができます。 例えば、

- **新規アプリケーション** matches **アプリケーション・ ニュース** 1, APPN.NEWLU, and so on
- APPN。 は、その LU 名に関係なく、ネットワーク名が **アタン**である任意の LU に一致します。
- **アタン** matches any LU with a network name beginning with **アタン**: **アプリケーション・ ニュース** 1, **アプリケーション・ ニュース** .LU2, and so on.

すべてのパートナー LU が同じリンクを使用してアクセスされるように、完全なワイルドカード・エントリーを指定するには、 ワイルドカード・ディスク を **類人猿** に設定し、 *fq_partner_lu* をヌル・ストリングに設定します。

ワイルドカード・ディスク
ワイルドカード・パートナー LU フラグ。 *fq_partner_lu* パラメーターに完全ワイルドカードまたは部分ワイルドカードが含まれているかどうかを示します。 可能な値は次のとおりです

類人猿
fq_partner_lu パラメーターにワイルドカード・エントリーが含まれています。

アプ・ ノー
fq_partner_lu パラメーターにワイルドカード・エントリーが含まれていません。

ls_name
ローカル LU (*lu_name* パラメーターによって指定される) とパートナー LU (*fq_partner_lu* パラメーターで指定) の間の通信に使用するリンク・ステーションの名前。 1 から 8 桁のローカル表示可能文字を指定します

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

ファイル名の変更
lu_name パラメーターに、無効な文字が含まれていました。

ファイル名の変更
fq_partner_lu パラメーターに無効な文字が含まれているか、または名前が完全修飾されていませんでした。

ファイル・ ワイルドカード名を使用できます
ワイルドカード・ディスク パラメーターが指定されましたが、 *fq_partner_lu* パラメーターが有効なワイルドカード名ではありませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: state check

If the verb does not execute because of a state error, CS Linux returns the following parameters.

primary_rc
AP_STATE_CHECK

secondary_rc
Possible values are:

AP_INVALID_LU_NAME
The local LU identified by the *lu_name* parameter does not exist.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_STATE_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

定義さるいタイムアウト

DEFINE_LU62_TIMEOUT verb 未使用の LU 6.2 セッションのタイムアウト期間を定義します。 各タイムアウトは、指定されたリソース・タイプとリソース名に対するものです。 既に定義されているリソース・タイプと名前のペアに対して DEFINE_* verb が発行された場合、このコマンドは以前の定義を上書きします。 新しいタイムアウト期間は、定義の変更後に活動化されたセッションでのみ使用されます。

1 つのセッションに対して複数の関連するタイムアウト期間が定義されている場合、最短期間が適用されます。

VCB 構造体

```
typedef struct define_lu62_timeout
{
  AP_UINT16         opcode;                     /* verb operation code   */
  unsigned char     reserv2;                    /* reserved              */
  unsigned char     format;                     /* reserved              */
  AP_UINT16         primary_rc;                 /* primary return code   */
  AP_UINT32         secondary_rc;               /* secondary return code */
  unsigned char     resource_type;              /* resource type         */
  unsigned char     resource_name[17];          /* resource name         */
  AP_UINT16         timeout;                    /* timeout               */
} DEFINE_LU62_TIMEOUT;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
AP_DEFINE_LU62_TIMEOUT タイムアウト

リソース・タイプ
定義するタイムアウトのタイプを指定します。 可能な値は次のとおりです

グローバル・ タイムアウトの追加
タイムアウトは、ローカル・ノードのすべての LU 6.2 セッションに適用されます。 リソース名 パラメーターは、すべてゼロに設定する必要があります。

ローカル・ タイムアウトの追加タイムアウト
タイムアウトは、 リソース名 パラメーターに指定されたローカル LU のすべての LU 6.2 セッションに適用されます。

パートナー LU_TIMEOUT の設定
タイムアウトは、 リソース名 パラメーターで指定されたパートナー LU へのすべての LU 6.2 セッションに適用されます。

AP_MODE_TIMEOUT
タイムアウトは、 リソース名 パラメーターに指定されたモードでのすべての LU 6.2 セッションに適用されます。

リソース名
照会されるリソースの名前。 この値は、以下のいずれかになります。

- リソース・タイプ が **グローバル・ タイムアウトの追加**に設定されている場合は、このパラメーターを指定しないでください。
- リソース・タイプ が **ローカル・ タイムアウトの追加タイムアウト**に設定されている場合は、ローカルの LU 名として、ローカルで表示可能なタイプ A の 1 から 8 文字を指定します。
- リソース・タイプ が **パートナー LU_TIMEOUT の設定**に設定されている場合は、以下のように、パートナー LU の完全修飾名を指定します。 17 のローカルで表示可能なタイプ A の文字は、 1 から 8 文字のネットワーク名、ピリオド、 1 から 8 文字のパートナー LU 名で構成されています。
- リソース・タイプ が AP_MODE_TIMEOUT に設定されている場合は、 1 から 8 文字のローカルで表示可能なタイプ A の文字をモード名として指定してください。

タイムアウト
タイムアウト期間 (秒)。 値 0 (ゼロ) は、セッションが即時に解放されることを示します。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

ソース・ リソース・ タイプのタイプ
定義されたタイムアウトのタイプが正しくありません。

ファイル名の変更
リソース・タイプ パラメーターに、無効な LU 名が指定されました。

初期化さができない
リソース・タイプ パラメーターに、無効なパートナー LU 名が指定されました。

ファイル・ パス名
リソース・タイプ パラメーターに、無効なモード名が指定されました。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

定義 LU_0_TO_3

DEFINE_LU_0_TO_3 verb は、 3270 エミュレーションまたは LUA (タイプ 0、 1、 2、または 3 の LU) を使用する LU を定義し、オプションで LU を LU プールに割り当てます。

この verb を使用して既存の LU を変更する場合は、 記述、 優先順位、および *lu_model* パラメーターのみを変更できます。その他のパラメーターはすべて、それらの既存の値に設定する必要があります。

VCB 構造体

```
typedef struct define_lu_0_to_3
{
    AP_UINT16           opcode;                 /* verb operation code       */
    unsigned char       reserv2;                /* reserved                  */
    unsigned char       format;                 /* reserved                  */
    AP_UINT16           primary_rc;             /* primary return code       */
    AP_UINT32           secondary_rc;           /* secondary return code     */
    unsigned char       lu_name[8];             /* LU name                   */
    LU_0_TO_3_DEF_DATA  def_data;               /* defined data              */
} DEFINE_LU_0_TO_3;
```

```
typedef struct lu_0_to_3_def_data
{
    unsigned char       description[32];         /* resource description      */
    unsigned char       reserv1[16];             /* reserved                  */
    unsigned char       nau_address;             /* LU NAU address            */
    unsigned char       pool_name[8];            /* LU Pool name              */
    unsigned char       pu_name[8];              /* PU name                   */
    unsigned char       priority;                /* LU priority               */
    unsigned char       lu_model;                /* LU model (type)           */
    unsigned char       sscp_id[6];              /* SSCP ID                   */
    AP_UINT16           timeout;                 /* Timeout                   */
    unsigned char       app_spec_def_data[16];   /* reserved                  */
    unsigned char       model_name[7];           /* reserved                  */
    unsigned char       term_method;             /* session termination type  */
    unsigned char       disconnect_on_unbind;    /* disconnect on UNBIND flag */
    unsigned char       reserv3[15];             /* reserved                  */
} LU_0_TO_3_DEF_DATA;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
LU_0_0_3 のアプ・デシン _3

lu_name
ローカル LU の名前。 これは、 8 バイトのタイプ A の EBCDIC ストリング (文字で始まる) で、右側に EBCDIC のスペースが埋め込まれます。

データの説明の説明
LU を記述するヌルで終了するテキスト・ストリング (0 から 31 文字の後にヌル文字を続けたストリング)。 このストリングは情報専用です。このストリングはノードの構成ファイルに保管され、 QUERY_LU_0_TO_3 verb で戻されますが、 CS Linux ではそれ以外の使用は行われません。

def_data.nau_address
LU のネットワーク・アクセス可能な装置アドレス。 これは、 1-255 の範囲内の数値です。

データ・プール名を定義してください
この LU が属しているプールの名前。 これは、タイプ A の EBCDIC ストリングで、名前が 8 バイトより短い場合は、右側に EBCDIC のスペースが埋め込まれます。 指定された名前のプールがまだ定義されていない場合、 CS Linux は、この名前を持つ新しいプールを追加し、そのプールに LU を割り当てます。

LU がプールに属していない場合は、このフィールドを 8 桁の 2 進ゼロに設定します。

データ *.pu_name* の定義
この LU が使用する PU の名前 (DEFINE_LS verb で指定されているもの)。 これは、 8 バイトのタイプ A の EBCDIC ストリング (文字で始まる) で、名前が 8 バイトより短い場合は、右側に EBCDIC のスペースが埋め込まれます。

データの優先順位
ホストへの送信時の LU の優先順位。 可能な値は次のとおりです

ネットワーク

上位 （ 上位 ）

メディア・メディア

低価格

データのデフラグ .lu_model
LU のタイプ。 可能な値は次のとおりです

モデル 2 の追加表示数

モデル 3 の最大表示数

モデルの追加 （ 表示モデル 4 ）

モデルの追加表示 （ 5 ）

プリンター・プリンター

AP_SCS_プリンタ

AP_RJE_WKSTN

不明 (ホストへのセッションが確立されると、 LU タイプが判別されます)

3270 エミュレーション用の LU を使用していない場合は、明示的な LU タイプを指定する必要はありません。このパラメーターを **不明**に設定してください。

指定する値に応じて、 CS Linux は、以下のいずれかのストリングを DDDLU NMVT 内のホストに送信し、標準 VTAM テーブルで使用される値と一致させます。

- 3270002 **モデル 2 の追加表示数** の場合
- 3270003 **モデル 3 の最大表示数** の場合

- 3270004 モデルの追加 （ 表示モデル 4 ）の場合
- 3270005 モデルの追加表示 （ 5 ）の場合
- 3270DSC プリンター・プリンター の場合
- 3270SCS AP_SCS_プリンタ の場合
- 3270000 AP_RJE_WKSTN の場合
- TN3270 クライアントを使用する 327000 *ん* 不明 。ここで、 ん はクライアントによって提供されるモデル番号 (2-5) です。
- LUA クライアントを使用する 不明 の 327000@

ホスト・システムが従属 LU (DDDLU) の動的定義をサポートしている場合、 CS Linux は、ホストへの通信リンクが確立されるときに、ホスト上で LU を動的に定義します。 TN3270 クライアントの場合は、このパラメーターを 不明に設定します。 次に、 CS Linux は、クライアントによって指定された端末タイプ (装置タイプ) から標準マッピングを使用して LU モデルを判別します。このマッピングを変更する必要がある場合は、 tn3270dev.dat （ 装置 ）ファイルの説明に従って *IBM Communications Server for Linux* 管理ガイド上のデータ・センター・デプロイメントを使用してこれを行うことができます。

ホストが DDDLU をサポートしていない場合は、ホスト構成に LU を組み込む必要があります。

データ .sscp_id
この LU の活動化を許可された SSCP の ID を指定します。 LU がどの SSCP によっても活動化できる場合は、このパラメーターを 0 (ゼロ) に設定します。 LU が特定の SSCP によってのみ活動化される場合は、このパラメーターの最初の 4 バイトを 0x05000000 に設定し、その LU の活動化を許可されている SSCP を識別する SSCP ID に最後の 2 バイトを設定します。

def_data.timeout
秒単位で指定された LU のタイムアウト。 タイムアウトがゼロ以外の値に設定されていて、 LU のユーザーがセッション非活動タイムアウトをサポートしている場合、指定された期間において PLU-SLU セッションが非アクティブのままで、かつ以下の条件のいずれかが存在すると、 LU は非活動化されます。

- セッションは限定リソース・リンクを介して渡されます。
- セッションが再び使用される前に、別のアプリケーションが LU を使用するように要求します。

タイムアウトが 0 (ゼロ) に設定されている場合、 LU は非活動化されません。

セッション非アクティブ・タイムアウトのサポートは、 LU を使用しているアプリケーション (3270 エミュレーション・プログラムなど) によって異なります。 LU が SNA ゲートウェイによって使用されている場合、セッション非アクティブ・タイムアウトは、 DEFINE_DOWNSTREAM_LU verb で *allow_timeout* が指定されている場合にのみサポートされます。

データの定義方法 *: term_method*
このパラメーターは、 CS Linux が、この LU からホストへの PLU-SLU セッションの終了を試行する方法を指定します。 可能な値は次のとおりです

ノード・ノードのデフォルト値を追加
DEFINE_NODE の *send_term_self* パラメーターで指定されたノードのデフォルトの終了方法を使用します。

未バインドの AP_SEND_UNBIND
UNBIND を送信して、セッションを終了します。

アプセンター M_SELF
TERM_SELF を送信して、セッションを終了します。

データを切断していません。非バインド
このパラメーターは、この LU が TN3270 クライアントによって使用されている場合のみ適用されます。 これは、ホストが VTAM MSG10 を表示する代わりに UNBIND を送信するか、またはホスト・セッション・マネージャーに戻るときに、セッションを終了するかどうかを指定します。 可能な値は次のとおりです

類人猿
ホストがタイプ 2 ではない UNBIND (BIND は着信) を送信した場合は、セッションを終了します。

アプ・ノー
ホストが UNBIND を送信した場合は、セッションを終了しないでください。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

ファイル名の変更
lu_name パラメーターに、無効な文字が含まれていました。

ファイル・プール名の変更
プール名 パラメーターに、無効な文字が含まれていました。

追加の無効メールアドレス
ナウド・アドレス パラメーターが許可された範囲内にありませんでした。

ファイルの無効優先度
優先順位 パラメーターが有効な値に設定されていませんでした。

メソッドの追加のメソッド
term_method パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc
可能な値は次のとおりです

エイプインバリデータプール名
プール名 パラメーターが無効でした。

付加が定義されていない
プール名 パラメーターが定義済みのどの PU 名とも一致しませんでした

エイプ無効 PU_TYPE
プール名 パラメーターで指定された PU は、ホスト PU ではありません。

マップ名前プールの名前が競合していない
LU 名は、 LU プールの名前と競合します。

追加の ALREADY_DEFINED
指定された名前の LU はすでに定義されています。

AP_LU_NAU_ADDR_ALREADY_DEFD
指定された NAU アドレスを持つ LU はすでに定義されています。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **状態検査の追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

目標の範囲が 0 から 3 までの範囲

DEFINE_LU_0_TO_3_RANGE verb は、 3270 エミュレーションまたは LUA (タイプ 0、 1、 2、または 3 の LU) で使用する LU の範囲を定義し、オプションで LU を LU プールに割り当てます。 この verb は、既存の LU の変更には使用できません

この verb に提供されるパラメーターには、新しい LU のベース名と NAU アドレスの範囲が含まれます。 新しい LU 名は、ベース名と NAU アドレスを結合することによって生成されます。 例えば、 LUNME のベース名と NAU 範囲の 11 から 14 を組み合わせると、 LU LUNME011、 LUNME012、 LUNME013 、および LUNME014 が定義されます。

VCB 構造体

```
typedef struct define_lu_0_to_3_range
{
     AP_UINT16        opcode;                 /* verb operation code       */
     unsigned char    reserv2;                /* reserved                  */
     unsigned char    format;                 /* reserved                  */
     AP_UINT16        primary_rc;             /* primary return code       */
     AP_UINT32        secondary_rc;           /* secondary return code     */
     unsigned char    base_name[6];           /* Base name                 */
     unsigned char    description[32];        /* resource description      */
     unsigned char    reserv1[16];            /* reserved                  */
     unsigned char    min_nau;                /* Minimum NAU address       */
     unsigned char    max_nau;                /* Maximum NAU address       */
     unsigned char    pool_name[8];           /* LU Pool name              */
     unsigned char    pu_name[8];             /* PU name                   */
     unsigned char    priority;               /* LU priority               */
     unsigned char    lu_model;               /* LU model (type)           */
     unsigned char    sscp_id[6];             /* SSCP ID                   */
     AP_UINT16        timeout;                /* Timeout                   */
     unsigned char    app_spec_def_data[16];  /* reserved                  */
     unsigned char    reserv3[7];             /* reserved                  */
     unsigned char    name_attributes;        /* Extension type            */
     unsigned char    base_number;            /* First extension number    */
     unsigned char    term_method;            /* session termination type  */
     unsigned char    disconnect_on_unbind;   /* disconnect on UNBIND flag */
     unsigned char    reserv4[13];            /* reserved                  */
} DEFINE_LU_0_TO_3_RANGE;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
LU_0_0_TO_3_RANGE の値を定義しています

基本名
新規 LU の名前のベース名。 これは 6 バイトのタイプ A の EBCDIC ストリング (文字で始まる) で、ベース名が 6 文字未満の場合は、右側に EBCDIC のスペースが埋め込まれます。

- 名前属性 パラメーターが **使用中の 16 進数名**に設定されている場合、この名前の長さは最大 6 文字にすることができます。 CS Linux は、この名前に 2 桁の 16 進数を付加することによって、各 LU の LU 名を生成します (基本番号 パラメーターで指定されたベース番号から開始します)。
- それ以外の場合、この名前の長さは 5 文字までです。 CS Linux は、この名前に 3 桁の 10 進数を追加することによって、各 LU の LU 名を生成します (NAU アドレスから取得されるか、 名前属性 パラメーターで指定された定義済みの基本番号から取得されます)。

記述
LU を記述するヌルで終了するテキスト・ストリング (0 から 31 文字の後にヌル文字を続けたもの)。 範囲内の各 LU には、同じストリングが使用されます。 このストリングは情報専用です。このストリングはノードの構成ファイルに保管され、 QUERY_LU_0_TO_3 verb で戻されますが、 CS Linux ではそれ以外の使用は行われません。

ミンナウ
1-255 の範囲内の最初の LU の NAU アドレス。

最大値
1-255 の範囲内の最後の LU の NAU アドレス。

プール名
これらの LU が属しているプールの名前。 これは 8 バイトのタイプ A の EBCDIC ストリングで、名前が 8 バイトより短い場合は、右側に EBCDIC のスペースが埋め込まれます。 指定された名前のプールがまだ定義されていない場合、 CS Linux は、この名前を持つ新しいプールを追加し、それに LU を割り当てます。

LU がプールに属していない場合は、このフィールドを 8 桁の 2 進ゼロに設定します。

プール名
これらの LU が使用する PU の名前 (DEFINE_LS verb で指定されているもの)。 これは、 8 バイトのタイプ A の EBCDIC ストリング (文字で始まる) で、右側に EBCDIC のスペースが埋め込まれます。

優先順位
ホストへの送信時の LU の優先順位。 可能な値は次のとおりです

ネットワーク

上位 （ 上位 ）

メディア・メディア

低価格

lu_model
LU のタイプ。 可能な値は次のとおりです

モデル 2 の追加表示数

モデル 3 の最大表示数

モデルの追加 （ 表示モデル 4 ）

モデルの追加表示 （ 5 ）

プリンター・プリンター

AP_SSCP_PRINTER

AP_RJE_WKSTN

不明 (ホストへのセッションが確立されると、 LU タイプが判別されます)

3270 エミュレーション用の LU を使用していない場合は、明示的な LU タイプを指定する必要はありません。このパラメーターを **不明**に設定してください。

指定する値に応じて、 CS Linux は、以下のいずれかのストリングを DDDLU NMVT 内のホストに送信し、標準 VTAM テーブルで使用される値と一致させます。

- 3270002 **モデル 2 の追加表示数** の場合
- 3270003 **モデル 3 の最大表示数** の場合
- 3270004 **モデルの追加 （ 表示モデル 4 ）**の場合
- 3270005 **モデルの追加表示 （ 5 ）**の場合
- 3270DSC **プリンター・プリンター** の場合
- 3270SCS AP_SCS_プリンタ の場合
- 3270000 AP_RJE_WKSTN の場合

- TN3270 クライアントを使用する 327000 *ん* 不明 。ここで、 ん はクライアントによって提供されるモデル番号 (2-5) です。
- LUA クライアントを使用する 不明 の 327000@

ホスト・システムが従属 LU (DDDLU) の動的定義をサポートしている場合、 CS Linux は、ホストへの通信リンクが確立されるときに、ホスト上で LU を動的に定義します。 TN3270 クライアントの場合は、このパラメーターを 不明に設定します。 次に、 CS Linux は、クライアントによって指定された端末タイプ (装置タイプ) から標準マッピングを使用して LU モデルを判別します。このマッピングを変更する必要がある場合は、 tn3270dev.dat （ 装置 ）ファイルの説明に従って *IBM Communications Server for Linux* 管理ガイド上のデータ・センター・デプロイメントを使用してこれを行うことができます。

ホストが DDDLU をサポートしていない場合、またはこのパラメーターが 不明に設定されている場合、 LU はホスト構成に含まれている必要があります。

sscp_id
この LU の活動化を許可された SSCP の ID を指定します。 0-65,535 の範囲の値を指定してください。 このパラメーターが 0 (ゼロ) に設定されている場合、 LU はどの SSCP によっても活動化できます。

タイムアウト
秒単位で指定された LU のタイムアウト。 タイムアウトがゼロ以外の値に設定されていて、 LU のユーザーがセッション非活動タイムアウトをサポートしている場合、指定された期間において PLU-SLU セッションが非アクティブのままで、かつ以下の条件のいずれかが存在すると、 LU は非活動化されます。

- セッションは限定リソース・リンクを介して渡されます。
- セッションが再び使用される前に、別のアプリケーションが LU を使用するように要求します。

タイムアウトが 0 (ゼロ) に設定されている場合、 LU は非活動化されません。

セッション非アクティブ・タイムアウトのサポートは、 LU を使用しているアプリケーション (3270 エミュレーション・プログラムなど) によって異なります。 LU が SNA ゲートウェイによって使用されている場合、セッション非アクティブ・タイムアウトは、 DEFINE_DOWNSTREAM_LU verb で *allow_timeout* が指定されている場合にのみサポートされます。

名前属性
定義される LU の属性。 可能な値は次のとおりです

追加なし
LU 名には、 NAU 番号に対応する番号があります。 数値は 10 進数で指定され、 基本名 パラメーターは 5 文字までしか指定できません。

平均 BASE_NUMBER
基本番号 パラメーターに指定されている値から、範囲内の LU の命名を開始します。

使用中の 16 進数名
10 進数ではなく 16 進数で、 LU 名に拡張子を追加してください。 基本名 パラメーターには、この値が指定されている場合は最大 6 文字を使用できます。

最大 10 進数の追加 （ _R ）
2 桁の 10 進数を使用して (このオプション 3 桁は使用せずに) LU 名に拡張子を追加します。 この属性を使用する場合は、この範囲内で最も 99 個の LU を定義できます。 基本名 パラメーターには、この値が指定されている場合は最大 6 文字を使用できます。

基本番号
名前属性 パラメーターに 平均 BASE_NUMBER が指定されている場合は、その範囲内の LU の命名を開始する番号を指定します。 この値は、 ミンナウ パラメーターの値の代わりに使用されます。

term_method
このパラメーターは、 CS Linux が、これらの LU の 1 つからホストへの PLU-SLU セッションを終了する方法を指定します。 可能な値は次のとおりです

ノード・ノードのデフォルト値を追加
DEFINE_NODE の *send_term_self* パラメーターで指定されたノードのデフォルトの終了方法を使用します。

未バインドの AP_SEND_UNBIND
UNBIND を送信して、セッションを終了します。

アプセンターター M_SELF
TERM_SELF を送信して、セッションを終了します。

切断されていない
このパラメーターは、この範囲の LU が TN3270 クライアントによって使用されている場合にのみ適用されます。 これは、ホストが VTAM MSG10 を表示する代わりに UNBIND を送信するか、またはホスト・セッション・マネージャーに戻るときに、セッションを終了するかどうかを指定します。 可能な値は次のとおりです

類人猿
ホストがタイプ 2 ではない UNBIND (BIND は着信) を送信した場合は、セッションを終了します。

アプ・ノー
ホストが UNBIND を送信した場合は、セッションを終了しないでください。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

ファイル名の変更
基本名 パラメーターに、無効な文字が含まれていました。

ファイル・プール名の変更
プール名 パラメーターに、無効な文字が含まれていました。

追加の無効メールアドレス
1 つ以上の LU アドレスが、許可された範囲内にありませんでした。

ファイルの無効優先度
優先順位 パラメーターが有効な値に設定されていませんでした。

メソッドの追加のメソッド
term_method パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc
可能な値は次のとおりです

エイプインバリデータプール名
プール名 パラメーターが無効でした。

付加が定義されていない
プール名 パラメーターが定義済みのどの PU 名とも一致しませんでした

エイプ無効 PU_TYPE
プール名 パラメーターで指定された PU は、ホスト PU ではありません。

マップ名前プールの名前が競合していない
範囲内の LU 名の 1 つが、 LU プールの名前と衝突しています。

追加の ALREADY_DEFINED
LU は、その範囲内のいずれかの LU の名前で既に定義されています。

AP_LU_NAU_ADDR_ALREADY_DEFD
LU は、その範囲内のいずれかの LU のアドレスを使用してすでに定義されています。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **状態検査の追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

DEFINE_LU_LU_PASSWORD

DEFINE_LU_LU_PASSWORD provides a password which is used for session-level security verification between a local LU and a partner LU.

VCB structure

```
typedef struct define_lu_lu_password
{
   AP_UINT16        opcode;                  /* verb operation code        */
   unsigned char    reserv2;                 /* reserved                   */
   unsigned char    format;                  /* reserved                   */
   AP_UINT16        primary_rc;              /* primary return code        */
   AP_UINT32        secondary_rc;            /* secondary return code      */
   unsigned char    lu_name[8];              /* local LU name              */
   unsigned char    lu_alias[8];             /* local LU alias             */
   unsigned char    fqplu_name[17];          /* fully qualified partner    */
                                             /*  LU name                   */
   unsigned char    verification_protocol;   /* verification protocol      */
   unsigned char    description[32];         /* resource description       */
   unsigned char    reserv1[16];             /* reserved                   */
   unsigned char    reserv3[8];              /* reserved                   */
   unsigned char    password[8];             /* password                   */
} DEFINE_LU_LU_PASSWORD;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DEFINE_LU_LU_PASSWORD

lu_name
LU name of the local LU, as defined to CS Linux. This is an 8-byte type-A EBCDIC string, padded on the right with spaces if the name is shorter than 8 bytes. To indicate that the LU is defined by its LU alias instead of its LU name, set this parameter to 8 binary zeros.

lu_alias
LU alias of the local LU, as defined to CS Linux. This is an 8-byte ASCII string, using any locally displayable characters, padded on the right with spaces if the name is shorter than 8 bytes. It is used only if *lu_name* is set to zeros.

To indicate the LU associated with the CP (the default LU), set both *lu_name* and *lu_alias* to 8 binary zeros.

fqplu_name
Fully qualified LU name for the partner LU, as defined to CS Linux. The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

verification_protocol
Requested LU-LU verification protocol to use. Possible values are:

AP_BASIC
Use basic LU-LU verification protocols.

AP_ENHANCED
Use enhanced LU-LU verification protocols.

AP_EITHER
Basic or enhanced verification is accepted.

description
A null-terminated text string (0-31 characters followed by a null character) describing the password. This string is for information only; it is stored in the node's configuration file and returned on the QUERY_LU_LU_PASSWORD verb, but CS Linux does not make any other use of it.

password
Password. This is an 8-byte hexadecimal string, which must not be set to all blanks or all zeros. It must match the equivalent parameter configured for the partner LU on the remote system (except that the least significant bit of each byte is not used in session-level security verification and does not need to match).

Whatever value the application supplies for this parameter is immediately replaced by the encrypted version of the password. Therefore, the value supplied for the *password* parameter is never written out.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

エイプ無効ルリエイリアス
lu_alias パラメーターが、定義済みの LU 別名と一致しませんでした。

ファイル名の変更
lu_name パラメーターが、定義されたローカル LU 名と一致しませんでした。

ファイル名の変更
fqplu_name パラメーターが、定義済みのパートナー LU 名と一致しませんでした。

パスワード無効パスワード
パスワード パラメーターがブランクまたはヌルです。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

確定 LU_POOL

この verb は、 LU プールを定義してそれに LU を割り当てるか、または既存のプールに LU を追加するために使用されます。 LU をプールに追加する前に、 LU を定義する必要があります。 LU を定義するときにプール名を指定して、プールを定義することもできます。詳細については、 129 ページの『定義 LU_0_TO_3』を参照してください。

この verb は、 LU を削除することによって既存のプールを変更するために使用することはできません。これを行うには、 DELETE_LU_POOL verb を使用します。

VCB structure

```
typedef struct define_lu_pool
{
     AP_UINT16          opcode;               /* verb operation code          */
     unsigned char      reserv2;              /* reserved                     */
     unsigned char      format;               /* reserved                     */
     AP_UINT16          primary_rc;           /* primary return code          */
     AP_UINT32          secondary_rc;         /* secondary return code        */
     unsigned char      pool_name[8];         /* LU pool name                 */
     unsigned char      description[32];      /* resource description         */
     unsigned char      reserv1[16];          /* reserved                     */
     unsigned char      reserv3[4];           /* reserved                     */
     AP_UINT16          num_lus;              /* number of LUs to add         */
     unsigned char      lu_names[10][8];      /* LU names                     */
} DEFINE_LU_POOL;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DEFINE_LU_POOL

pool_name
Name of the LU pool. This is an 8-byte type-A EBCDIC string, padded on the right with EBCDIC spaces if the name is shorter than 8 bytes. If a pool of this name is not already defined, CS Linux creates it.

description
A null-terminated text string (0-31 characters followed by a null character) describing the pool. This string is for information only; it is stored in the node's configuration file and returned on the QUERY_LU_POOL verb, but CS Linux does not make any other use of it.

num_lus
Number of LUs to be added to the pool. This can be zero to define the pool without adding any LUs, or 1-10. To create a pool containing more than 10 LUs, issue multiple DEFINE_LU_POOL verbs for the same pool name.

lu_names
Names of the LUs that are being assigned to the pool. Each of these LUs must already be defined to CS Linux as an LU of type 0-3. Each LU name is an 8-byte type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces.

If a specified LU is currently assigned to a different pool, CS 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 verb.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

ファイル名の変更
指定された LU 名の 1 つ以上が、定義済みの LU 名と一致しませんでした。

ファイル・ プール名の変更
プール名 パラメーターに、無効な文字が含まれていました。

類型の無効です。
無感覚 パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: state check

If the verb does not execute because of a state error, CS Linux returns the following parameters.

primary_rc
AP_STATE_CHECK

secondary_rc

AP_LU_NAME_POOL_NAME_CLASH
The specified pool name clashes with the name of an LU.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_STATE_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

定義モード

DEFINE_MODE verb は、モード (セッションのグループが使用するネットワーク特性のセット) を定義するか、以前に定義されたモードを変更します。 SNA 定義モード CPSVCMG 変更したり、 SNA 定義モード **スナスフコマン**で使用される COS 名を変更することはできません。

この verb を使用して既存のモードを変更すると、変更内容は、変更を行った後にモードの使用を開始するローカル LU とパートナー LU の新しい組み合わせに適用されます。 ただし、既にモードを使用している LU の組み合わせによっては、次のローカルまたはリモートから開始された CNOS コマンドの後まで変更が反映されません。

また、この verb を使用して、認識されないモードがマップされるデフォルトの COS を指定することもできます。 デフォルトの COS が指定されていない場合は、 SNA 定義の COS #CONNECT が使用されます。

VCB structure

```
typedef struct define_mode
{
  AP_UINT16       opcode;                /* verb operation code            */
  unsigned char   reserv2;               /* reserved                       */
  unsigned char   format;                /* reserved                       */
```

```
  AP_UINT16       primary_rc;             /* primary return code            */
  AP_UINT32       secondary_rc;           /* secondary return code          */
  unsigned char   mode_name[8];           /* mode name                      */
  AP_UINT16       reserv3;                /* reserved                       */
  MODE_CHARS      mode_chars;             /* mode characteristics           */
} DEFINE_MODE;
```

```
typedef struct mode_chars
{
  unsigned char   description[32];        /* resource description           */
  unsigned char   reserv2[16];            /* reserved                       */
  AP_UINT16       max_ru_size_upp;        /* maximum RU size upper bound    */
  unsigned char   receive_pacing_win;     /* receive pacing window          */
  unsigned char   default_ru_size;        /* default RU size to maximize    */
                                          /*  performance                   */
  AP_UINT16       max_neg_sess_lim;       /* maximum negotiable session limit*/
  AP_UINT16       plu_mode_session_limit; /* LU-mode session limit          */
  AP_UINT16       min_conwin_src;         /* minimum source contention winner*/
                                          /*  sessions                      */
  unsigned char   cos_name[8];            /* class of service name          */
  unsigned char   cryptography;           /* reserved                       */
  unsigned char   compression;            /* data compression supported?    */
  AP_UINT16       auto_act;               /* initial auto-activation count  */
  AP_UINT16       min_conloser_src;       /* min source contention loser    */
  AP_UINT16       max_ru_size_low;        /* maximum RU size lower bound    */
  AP_UINT16       max_receive_pacing_win; /* maximum receive pacing window  */
  unsigned char   max_compress_lvl;       /* max level of data compression  */
  unsigned char   max_decompress_lvl;     /* max level of data decompression */
  unsigned char   comp_in_series;         /* reserved                       */
  unsigned char   reserv4[25];            /* reserved                       */
} MODE_CHARS;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DEFINE_MODE

mode_name
Name of the mode. This is an 8-byte type-A EBCDIC string, padded on the right with EBCDIC spaces if the name is shorter than 8 bytes. The name must start with a letter, or can start with # for one of the SNA-defined modes such as #INTER. For information about SNA-defined modes, see the *IBM Communications Server for Data Center Deployment on Linux Administration Guide*.

To specify the default COS that will be used for any unrecognized mode names, set this parameter to 8 binary zeros. In this case, the *mode_chars.cos_name* parameter is taken as the default COS name; all other parameters supplied on this verb are ignored.

mode_chars.description
A null-terminated text string (0-31 characters followed by a null character) describing the mode. This string is for information only; it is stored in the node's configuration file and returned on the QUERY_MODE_DEFINITION and QUERY_MODE verbs, but CS Linux does not make any other use of it.

mode_chars.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.

Range: 256-61,440. If the *default_ru_size* parameter (see below) is set to AP_YES, this parameter is ignored (and the value is not checked).

mode_chars.receive_pacing_win
Session pacing window for sessions using this mode; the range is 1-63. This value is used only for fixed pacing (not for adaptive pacing), and specifies the maximum number of frames that can be received from the partner LU before the local LU must send a response. CS Linux always uses adaptive pacing unless the adjacent node specifies that it is not supported.

mode_chars.default_ru_size
Specifies whether a default upper bound for the maximum RU size will be used. Possible values are:

AP_YES
CS Linux ignores the *max_ru_size_upp* parameter, and sets the upper bound for the maximum RU size to the largest value that can be accommodated in the link BTU size.

AP_NO
CS Linux uses the *max_ru_size_upp* parameter to define the maximum RU size.

mode_chars.max_neg_sess_lim
Maximum number of sessions allowed on this mode between any local LU and partner LU. This value may be lowered for a particular LU-LU-mode combination when issuing *initialize_session_limit* or *change_session_limit*.

Range: 1-32,767. Zero indicates that CS 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.

mode_chars.plu_mode_session_limit
Default session limit for this mode. This 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 1-32,767 (which must not exceed the value in *max_neg_sess_lim*). Zero indicates that CS 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.

mode_chars.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 CS 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*.

mode_chars.cos_name
Name of the class of service to request when activating sessions on this mode.

If the node supports mode to COS mapping (as defined by the *mode_to_cos_map_supp* parameter on DEFINE_NODE), the COS specified by this field must be either an SNA defined COS or a COS previously defined by issuing a DEFINE_COS verb. Otherwise, this parameter is ignored.

The name is an 8-byte type-A character string, padded on the right with spaces if the name is shorter than 8 characters.

mode_chars.compression
Specifies whether sessions activated using this mode can use compression. Possible values are:

AP_COMP_PROHIBITED
Compression is not supported for sessions using this mode.

AP_COMP_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.)

mode_chars.auto_act
Number of sessions that will be activated automatically for 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.

mode_chars.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*.

mode_chars.max_ru_size_low
Lower bound for the maximum size of RUs sent and received on sessions that use this mode. Specify a value in the range 256-61,440. The value 0 means that there is no lower bound.

The value is used when the maximum RU size is negotiated during session activation. This parameter is ignored if the *default_ru_size* parameter is set to AP_YES.

mode_chars.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. (CS 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. The value zero means that there is no upper bound.

mode_chars.max_compress_lvl
Specifies the maximum level of compression that CS Linux will attempt to negotiate for data flowing from the local node. Possible values are:

- AP_NONE
- AP_RLE_COMPRESSION
- AP_LZ9_COMPRESSION
- AP_LZ10_COMPRESSION

If compression is negotiated using a non-extended BIND, which does not specify a maximum compression level, RLE compression will be used.

mode_chars.max_decompress_lvl
Specifies the maximum level of decompression that CS Linux will attempt to negotiate for data flowing into the local node. Possible values are:

- AP_NONE
- AP_RLE_COMPRESSION
- AP_LZ9_COMPRESSION
- AP_LZ10_COMPRESSION

If compression is negotiated using a non-extended BIND, which does not specify a maximum compression level, RLE compression will be used.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

付加 CPSVCMG_ALREADY_DEFD
SNA 定義モード CPSVCMG を変更することはできません。

AP_INVALID_CNOS_SLIM
plu_mode_session_limit パラメーターが無効です。

ファイルの無効圧縮
圧縮 パラメーターが無効です。

無効なファイル名の変更
cos_name パラメーターが、定義済みの COS 名と一致しませんでした。

ファイル・コード・スナップショット・モードの追加モード
SNA 定義モードの スナスフコマン の COS を変更することはできません。

AP_INVALID_DEFAULT_RU_SIZE
default_ru_size パラメーターが有効範囲内にありませんでした。

エージェントの最大値を入力している
max_compress_lvl パラメーターが無効です。

最大圧縮解除 LVL
最大圧縮 *lvl* パラメーターが無効です。

エージェント・ファイルに占有されている場合は、そのファイルには以下のようにします。
最大値の *sess_lim* パラメーターが有効範囲内にありませんでした。

ファイルの最大サイズが無効です
最大 *ru_size_upp* パラメーターが有効範囲内にありませんでした。

非バリデータ・コンローサー
min_conloser_src パラメーターが有効な範囲内になかったか、または *plu_mode_session_limit* より大きくなっていました。

未検証 MIN_CON 勝者
min_conwin_src パラメーターが有効な範囲内になかったか、または *plu_mode_session_limit* より大きくなっていました。

未検証の最小競合合計
min_conwin_src パラメーターと *min_conloser_src* パラメーターの合計が *plu_mode_session_limit* より大きくなっていました。

ファイル・パス名
モード名 パラメーターに、無効な文字が含まれていました。

「ファイル・バリデータの追加」ウィンドウ
受信パチンイン パラメーターが有効範囲内にありませんでした。

ファイル・モードの追加のモードでの値の制限
SNA 定義モードの スナスフコマン は、 2 と 1 *min_conwin_src* セッション限度を持っている必要があります。 これらのパラメーターに異なる値を使用して スナスフコマン を定義することは

アプリケーション・ SESS_LIM_EXCEEDS_NEG
plu_mode_session_limit に指定された値が、 最大値の *sess_lim* に指定された値より大きくなっていました。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

DEFINE_NODE

An application issues this verb in order to define a new node, or to modify the parameters of an inactive node.

This verb must be issued to a server where the node is not running. It cannot be issued to a running node.

VCB structure

```
typedef struct define_node
{
  AP_UINT16         opcode;             /* verb operation code              */
  unsigned char     reserv2;            /* reserved                         */
  unsigned char     format;             /* reserved                         */
  AP_UINT16         primary_rc;         /* primary return code              */
  AP_UINT32         secondary_rc;       /* secondary return code            */
  unsigned char     node_name[128];     /* name of Node                     */
  AP_UINT32         target_handle;      /* handle for subsequent verbs      */
  CP_CREATE_PARMS   cp_create_parms;    /* CP create parameters             */
} DEFINE_NODE;
```

```
typedef struct cp_create_parms
{
  AP_UINT16       crt_parms_len;        /* length of CP_CREATE_PARMS        */
  unsigned char   description[32];       /* resource description            */
  unsigned char   reserv1[2];           /* reserved                         */
  unsigned char   ms_support;           /* reserved                         */
  unsigned char   queue_nmvts;          /* reserved                         */
  unsigned char   reserv3[12];          /* reserved                         */
  unsigned char   node_type;            /* node type                        */
  unsigned char   fqcp_name[17];        /* fully qualified CP name          */
  unsigned char   cp_alias[8];          /* CP alias                         */
  unsigned char   mode_to_cos_map_supp; /* mode to COS mapping support      */
  unsigned char   mds_supported;        /* MDS and MS capabilities          */
  unsigned char   node_id[4];           /* node ID                          */
  AP_UINT16       max_locates;          /* maximum locates node can process */
  AP_UINT16       dir_cache_size;       /* directory cache size (reserved   */
                                        /*  is not NN)                      */
  AP_UINT16       max_dir_entries;      /* maximum directory entries (zero  */
                                        /*  means unlimited)                */
  AP_UINT16       locate_timeout;       /* locate timeout in seconds        */
  unsigned char   reg_with_nn;          /* register resources with NNS      */
  unsigned char   reg_with_cds;         /* register resources with CDS      */
  AP_UINT16       mds_send_alert_q_size;/* size of MDS send alert queue     */
  AP_UINT16       cos_cache_size;       /* number of cos definitions        */
  AP_UINT16       tree_cache_size;      /* Topology Database routing tree   */
                                        /*  cache size                      */
  AP_UINT16       tree_cache_use_limit; /* number of times a tree can be used*/
  AP_UINT16       max_tdm_nodes;        /* max number of nodes that can be  */
                                        /*  stored in Topology Database     */
  AP_UINT16       max_tdm_tgs;          /* max number of TGs that can be    */
                                        /*  stored in Topology Database     */
  AP_UINT32       max_isr_sessions;     /* maximum ISR sessions             */
  AP_UINT32       isr_sessions_upper_threshold;  /* upper threshold for ISR  */
                                                 /*  sessions                */
  AP_UINT32       isr_sessions_lower_threshold;  /* lower threshold for ISR  */
                                                 /*  sessions                */
  AP_UINT16       isr_max_ru_size;      /* max RU size for ISR              */
  AP_UINT16       isr_rcv_pac_window;   /* ISR receive pacing window size   */
  unsigned char   store_endpt_rscvs;    /* endpoint RSCV storage            */
  unsigned char   store_isr_rscvs;      /* ISR RSCV storage                 */
  unsigned char   store_dlur_rscvs;     /* DLUR RSCV storage                */
  unsigned char   dlur_support;         /* is DLUR supported?               */
  unsigned char   pu_conc_support;      /* is PU conc supported?            */
  unsigned char   nn_rar;               /* Route additional resistance      */
  unsigned char   hpr_support;          /* Level of hpr support             */
  unsigned char   mobile;                  /* reserved                      */
  unsigned char   discovery_support;       /* reserved                      */
  unsigned char   discovery_group_name[8]; /* reserved                      */
  unsigned char   implicit_lu_0_to_3;      /* reserved                      */
  unsigned char   default_preference;      /* reserved                      */
  unsigned char   anynet_supported;        /* reserved                      */
  AP_UINT16       max_ls_exception_events; /* Max # exception entries       */
  unsigned char   comp_in_series;          /* reserved                      */
  unsigned char   max_compress_lvl;        /* reserved                      */
  unsigned char   node_spec_data_len;      /* reserved                      */
  unsigned char   ptf[64];                 /* program temporary fix array   */
  unsigned char   cos_table_version;       /* version of COS tables to use  */
  unsigned char   send_term_self;          /* default PLU-SLU session term  */
  unsigned char   disable_branch_awareness; /* disable BrNN awareness       */
  unsigned char   cplu_syncpt_support;     /* syncpoint support on CP LU?   */
  unsigned char   cplu_attributes;         /* attributes for CP LU          */
```

```
  unsigned char   reserved[95];           /* reserved                        */
} CP_CREATE_PARMS;
```

提供されるパラメーター

オペコード
アプリケーションの DEFINE_NODE

ノード名
アプリケーションが定義しようとしている CS Linux ノードの名前。

ノード名に . (ピリオド) 文字が含まれている場合、 CS Linux は、ノード名が完全修飾名であると見なします。それ以外の場合は、 DNS ルックアップを実行してノード名を判別します。

cp_create_parms.crt_parms_len
作成パラメーター構造の長さ。

cp_create_parms.description
ノードを記述するテキスト・ストリング (0 から 31 文字の後にヌル文字が続く) 。 このストリングは情報専用です。このストリングはノードの構成ファイルに保管され、 QUERY_NODE verb で戻されますが、 CS Linux ではそれ以外の使用は行われません。

cp_create_parms.node_type
以下のノード・タイプのいずれか。

AP_NETWORK_NODE

AP_BRANCH_NETWORK_NODE

ノードの追加 （ _L ）

AP_LEN_NODE

cp_create_parms.fqcp_name
ノードの完全修飾 CP 名。 この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

cp_create_parms.cp_alias
ローカルで使用される CP の別名。 この別名は、 APPC アプリケーションが CP LU にアクセスするために使用することができます。 これは 8 バイトの ASCII ストリングです。 8 バイトすべてが重要であり、設定する必要があります。

cp_create_parms.mode_to_cos_map_supp
ノードがモードから COS へのマッピングを行うかどうかを指定します。 このパラメーターは、ネットワーク・ノードでは無視されます。 mode - to-COS マッピングは常にサポートされます。 LEN ノードの場合、モードから COS へのマッピングはサポートされていません。 可能な値は次のとおりです

類人猿
モードから COS へのマッピングがサポートされています。 このノードに定義されたモードには、 SNA 定義の COS または DEFINE_COS を使用して定義された COS 名のいずれかを指定する、関連する COS 名を含める必要があります。

アプ・ノー
モードから COS へのマッピングはサポートされていません。 デフォルトの COS 名が使用されます。

サポートされる *cp_create_parms.mds__*
管理サービスが複数ドメイン・サポートと MS ケイパビリティーをサポートするかどうかを 指定 可能な値は次のとおりです

類人猿
MDS がサポートされている

アプ・ノー
MDS はサポートされません。

cp_create_parms.node_id
XID 交換で使用されるノード ID 。 これは 4 バイトの 16 進数ストリングで、ブロック番号 (3 桁の 16 進数字) とノード番号 (5 桁の 16 進数字) から構成されます。

cp_create_parms.max_locates
ノードが同時に処理できる位置指定要求の最大数。 未処理の位置指定要求 (応答がまだ受信されていない要求) の数がこの限度に達すると、それ以上の位置指定要求はリジェクトされます。 最小値は 8 です。

cp_create_parms.dir_cache_size
ネットワーク・ノードのみ: ディレクトリー・キャッシュのサイズ。 最小値は 3 です。 QUERY_DIRECTORY_STATS で戻される情報を使用して、適切なサイズの判別に役立たせることができます。

cp_create_parms.max_dir_entries
ディレクトリー項目の最大数。 制限なしでゼロを指定してください。

cp_create_parms.locate_timeout
ネットワーク検索がタイムアウトになるまでの時間を秒単位で指定します。 タイムアウトなしでゼロを指定してください。

cp_create_parms.reg_with_nn
ノードが開始されたときに、ノードのリソースをネットワーク・ノード・サーバーに登録するかどうかを指定します。 このパラメーターの有効な値は、ノードがエンド・ノードであるか、分岐ネットワーク・ノードであるかによって異なります。ローカル・ノードがネットワーク・ノードである場合、または LEN ノードである場合、このパラメーターは予約されます。

終了ノードの可能な値 :

類人猿
リソースを NN に登録します。 エンド・ノードのネットワーク・ノード・サーバーは、宛先指定された場所のみを転送します。

アプ・ノー
リソースを登録しません。 ネットワーク・ノード・サーバーは、すべてのブロードキャスト検索をエンド・ノードに転送します。

分岐ネットワーク・ノードの可能な値 :

追加レジストリーなし （ _C ）
ローカル・ノードは、 NN サーバーと一緒に LU を登録しません。 NN サーバーは、すべてのブロードキャスト検索をブランチ・ネットワーク・ノードに転送します。

すべての登録登録 （ _R ）
ローカル・ノードは、すべてのドメイン独立 LU を NN サーバーに登録します。これは、 NN サーバーがオプション・セット 1116 をサポートする場合は、すべてのローカル従属 LU も登録します。 NN サーバーは、 (登録できなかった従属 LU を所有している場合を除いて) 、宛先指定された場所のみを転送します。

登録領域のローカルのみ （ _R ）
ローカル・ノードは、すべてのローカル独立 LU を NN サーバーに登録します。これは、 NN サーバーがオプション・セット 1116 をサポートする場合は、すべてのローカル従属 LU も登録します。 NN サーバーは、すべてのブロードキャスト検索をブランチ・ネットワーク・ノードに転送します。

パラメーターの作成中に *parms.reg_with_cds* を返します。
エンド・ノード: ネットワーク・ノード・サーバーが、エンド・ノード・リソースをセントラル・ディレクトリー・サーバーに登録できるかどうかを指定します。 *reg_with_nn* を **アプ・ノー**に設定すると、このフィールドは無視されます。

ネットワーク・ノード: ローカル・リソースまたはドメイン・リソースをオプションでセントラル・ディレクトリー・サーバーに登録できるかどうかを指定

可能な値は次のとおりです

類人猿
CDS にリソースを登録します。

アプ・ノー
リソースを登録しません。

分岐ネットワーク・ノード: ネットワーク・サーバーによって、 BrNN リソース (分岐ネットワーク・ノードまたはそのドメインのローカル・リソース) を中央ディレクトリー・サーバーに登録できるかどうかを指定します。 *reg_with_nn* を **追加レジストリーなし** (_C) に設定すると、このフィールドは無視されます。

可能な値は次のとおりです

類人猿
CDS にリソースを登録します。

アプ・ノー
リソースを登録しません。

cp_create_parms.mds_send_alert_q_size
MDS 送信アラート・キューのサイズ。 キューに入れられたアラートの数がこの制限に達すると、 CS Linux はキュー上の最も古いアラートを削除します。 CS Linux は、大きい数を指定しない限り、 2 の値を使用します

cp_create_parms.cos_cache_size
COS データベース・ウェイト・キャッシュのサイズ。 この値は、必要な COS 定義の最大数に設定する必要があります。 CS Linux は、大きい数を指定しない限り、 8 の値を使用します

cp_create_parms.tree_cache_size
ネットワーク・ノード: トポロジー・データベース・ルーティング・ツリー・キャッシュ・サイズのサイズ。 最小値は 8 です。 エンド・ノードまたは LEN ノードの場合、このパラメーターは予約されています。

cp_create_parms.tree_cache_use_limit
ネットワーク・ノード: キャッシュ・ツリーの使用の最大数。 この数を超えると、ツリーは破棄されて再計算されます。 これにより、ノードは均等な重み経路間でのセッションのバランスを取る 低い値を指定すると、活動化の待ち時間が長くなるとロード・バランシングが向上する 最小値は 1 です。 エンド・ノードまたは LEN ノードの場合、このパラメーターは予約されています。

cp_create_parms.max_tdm_nodes
ネットワーク・ノード: トポロジー・データベースに保管できるノードの最大数 (ゼロは無制限)。 エンド・ノードまたは LEN ノードの場合、このパラメーターは予約されています。

cp_create_parms.max_tdm_tgs
ネットワーク・ノード: トポロジー・データベースに保管できる TG の最大数 (ゼロは無制限)。 エンド・ノードまたは LEN ノードの場合、このパラメーターは予約されています。

cp_create_parms.max_isr_sessions
ネットワーク・ノード: ノードが一度に参加できる ISR セッションの最大数。 より大きな数を指定しない限り、 CS Linux は値 100 を使用します。 エンド・ノードまたは LEN ノードの場合、このパラメーターは予約されています。

cp_create_parms.isr_sessions_upper_threshold*、 *cp_create_parms.isr_sessions_lower_threshold
ネットワーク・ノード: これらのしきい値は、経路計算で使用するためにネットワーク内の他のノードに報告される、ノードの輻輳状況を制御します。 ISR セッションの数が上限しきい値を超える場合、ノード状態は輻輳していない状態から輻輳に変わります。 ISR セッションの数が下限しきい値を下回ると、ノードの状態は未輻輳に戻ります。 下限しきい値は上限しきい値より小さくなければならず、上限しきい値は 最大 *isr_sessions* より小さい値でなければなりません。 エンド・ノードまたは LEN ノードの場合、このパラメーターは予約されています。

cp_create_parms.isr_max_ru_size
ネットワーク・ノードまたは BrNN: 中間セッションまたは DLUR LU-LU セッションでサポートされる最大 RU サイズ。 指定された値が有効な RU サイズではない場合 (システム・ネットワーク体系*:* フォーマットで説明されているように) 、 CS Linux はこの値を次の有効な値に切り上げます。

エンド・ノード: DLUR LU-LU セッションでサポートされる最大 RU サイズ。

LEN ノードの場合、このパラメーターは予約されています。

cp_create_parms.isr_rcv_pac_window
ネットワーク・ノード: 中間セッションの受信ペーシング・ウィンドウ・サイズ (1-63 の範囲) を推奨します。 この値は、隣接ノードが適応ペーシングをサポートしていない場合に、中間セッションの 2 次ホップでのみ使用されます。 エンド・ノードまたは LEN ノードの場合、このパラメーターは予約されています。

cp_create_parms.store_endpt_rscvs
診断の目的で RSCV を保管する必要があるかどうかを指定します (**類人猿** または **アプ・ノー**)。 このフィールドが **類人猿**に設定されている場合は、 QUERY_SESSION verb で RSCV が戻されます。 (この値を **類人猿** に設定すると、エンドポイント・セッションごとに RSCV が保管されます。 この追加ストレージは、 1 セッション当たり最大 256 バイトにすることができます。)

cp_create_parms.store_isr_rscvs
ネットワーク・ノード: 診断目的のために RSCV を保管するかどうかを指定します (**類人猿** または **アプ・ノー**)。 このフィールドが **類人猿**に設定されている場合は、 QUERY_ISR_SESSION verb で RSCV が戻されます。 (この値を **類人猿** に設定すると、 ISR セッションごとに RSCV が保管されます。 この追加ストレージは、 1 セッション当たり最大 256 バイトにすることができます。) エンド・ノードまたは LEN ノードの場合、このパラメーターは予約されています。

cp_create_parms.store_dlur_rscvs
診断の目的で RSCV を保管する必要があるかどうかを指定します (**類人猿** または **アプ・ノー**)。 このフィールドが **類人猿**に設定されている場合は、 QUERY_DLUR_LU verb で RSCV が戻されます。 (この値を **類人猿** に設定すると、 DLUR を使用する各 PLU-SLU セッションに対して、 RSCV が保管されます。 この追加ストレージは、 1 セッション当たり最大 256 バイトにすることができます。)

cp_create_parms.dlur_support
DLUR をサポートするかどうかを指定 LEN ノードの場合、このパラメーターは予約されています。 可能な値は次のとおりです

類人猿
DLUR がサポートされている

無制限に追加された DLUR_MULTI_サブネット | AP_YES
エンド・ノードまたは分岐ネットワーク・ノード: DLUR はサポートされていますが、別のサブネット内の DLUS に接続するために使用されません。 マルチサブネット操作が不要な場合は、 AP_YES の代わりにこの値を使用して、ネットワーク・ノードでのネットワーク・トラフィックと輻輳を軽減する必要があります。

この値は、ネットワーク・ノードではサポートされません。

アプ・ノー
DLUR はサポートされません。

cp_create_parms.pu_conc_support
SNA ゲートウェイをサポートするかどうかを指定します (**類人猿** または **アプ・ノー**)。

ノードがダウンストリーム LU と通信する 1 次 RUI アプリケーションを実行するために使用される場合、このパラメーターは **類人猿**に設定する必要があります。

キュー内のパラメーター *.nn_rar*
0-255 範囲内のネットワーク・ノードの経路追加抵抗値。

cp_create_parms.hpr_support
ノードによって提供される HPR (高性能経路指定) サポートのレベルを指定します。 可能な値は次のとおりです

追加なし
HPR はサポートされない。

ベース・ベース
このノードは、自動ネットワーク経路指定 (ANR) を実行できますが、 HPR セッションの RTP (高速トランスポート・プロトコル) 終了点として機能することはできません。

追加 (_R)
このノードは、自動ネットワーク経路指定 (ANR) を実行でき、 HPR セッションの RTP (高速トランスポート・プロトコル) 終了点として機能することができます。

追加の制御フロー
このノードは制御フローを含むすべての HPR 機能を実行できます。

ローカル・ノードが LEN ノードである場合、このパラメーターは **追加なし**に設定する必要があります。 それ以外の場合、推奨設定は **追加の制御フロー**です

このノードで Enterprise Extender (HPR/IP) または MPC+ リンクを使用している場合は、このパラメーターを **追加の制御フロー**に設定する必要があります。

cp_create_parms.max_ls_exception_events
ノードによって記録される LS 例外イベントの最大数。

cp_create_parms.ptf
今後のプログラム一時修正 (ptf) 操作を構成および制御するための配列は、以下のとおりです。

cp_create_parms.ptf[0]
REQDISCONT サポートおよび必須検索状況サポート。

CS Linux は通常、 REQDISCONT を使用して、セッション・トラフィックに必要とされなくなった限定リソース・ホスト・リンクを非活動化します このバイトは、 REQDISCONT の使用を抑止するため、または CS Linux によって送信された REQDISCONT 要求で使用される設定値を変更するために使用できます。 このバイトは、以下のいずれかの値に設定します。

追加なし
通常の REQDISCONT サポートを使用してください。

追加抑制要求の却下
REQDISCONT を使用しないでください。

オーバーライド・ REQDISCONT
変更されたバージョンの REQDISCONT サポートを使用してください。 REQDISCONT を指定する場合は、論理 **それとも** 操作を使用して、以下の値の 1 つまたは両方と結合させる必要があります。

追加要求のタイプの追加
REQDISCONT にタイプ "即時" を使用します。この値が指定されていない場合、 CS Linux はタイプ "正常"を使用します。

REQDISCONT_RE コンタクト
REQDISCONT にタイプ "即時再連絡" を使用します。この値が指定されていない場合、 CS Linux はタイプ "即時の再連絡なし"を使用します。

CS Linux は、エンド・ノードとして、または分岐ネットワーク・ノードとして実行されている場合、ネットワーク検索をネットワーク・ノード・サーバー (NNS) から招待するかどうかを選択することができます。 ネットワーク検索を要求すると、ネットワーク全体のブロードキャスト検索処理が遅くなるため、望ましくない場合があります。 ただし、ローカル・ノードがそのすべてのリソース (LU) をその NNS に登録できない場合、検索を要求する方法は、これらのリソースをネットワークから可視にする唯一の方法です。

通常、 CS Linux は、すべての LU を登録できるかどうかを判別し、その NNS からネットワーク検索をインテリジェントに要求します。 このノードが、ネットワークが異常な方法でネットワークにアクセスできるようにする場合 (例えば、他のノードのゲートウェイとして機能している場合) 、上記の値を以下の値と結合して、標準操作をオーバーライドします。

追加セット検索状況
NNS からのネットワーク検索を無条件に要求します。

cp_create_parms.ptf[1]
ERP サポート。 CS Linux は通常、 ERP として ACTPU(ERP) を処理します。これにより、 PU-SSCP セッションはリセットされますが、従属する LU-SSCP セッションと PLU-SLU セッションは暗黙的に非活動化されません。 SNA 実装は、 ACTPU(コールド)であるかのように ACTPU(ERP) を法的に処理することができ、従属する LU-SSCP セッションと PLU-SLU セッションを暗黙的に非活動化します。 このバイトは、以下のいずれかの値に設定します。

追加なし
通常処理を使用します。

オーバーライド ERP の指定
すべての ACTPU 要求を ACTPU(コールド)として処理します。

cp_create_parms.ptf[2]
LU 6.2 セッションの活動化および非活動化。 CS Linux は通常、従属 LU 6.2 セッションを活動化するときに INIT_SELF メッセージに ENQUEUE パラメーターを組み込みません。また、限定リソース LU 6.2 セッションを非活動化する前に BIS プロトコルを使用します。

通常の処理を使用するには、このパラメーターを **追加なし**に設定します。

LU 6.2 セッションの活動化および非活動化をカスタマイズするには、次のいずれかの値を指定します

抑止ビンの追加
BIS プロトコルは使用しないでください。 限定リソース LU 6.2 セッションは、 UNBIND(クリーンアップ)を使用して即時に非活動化されます

AP_LU62_INIT_SELF_ENQUEUE
ENQUEUE パラメーターを含む INIT_SELF メッセージの古いフォーマットを使用します。

cp_create_parms.ptf[3]
APINGD サポート。 CS Linux には通常、 APING 接続テスターのパートナー・プログラムが含まれています。 このバイトを使用すると、ノード内の APING デーモンを使用不可にして、ノードに到着する APING プログラムによる要求が自動的に処理されないようにすることができます。 このバイトは、以下のいずれかの値に設定します。

追加なし
ノード内に APINGD サポートを組み込みます (通常の処理)。

外部外部データの追加
ノード内で APINGD を無効にします。

cp_create_parms.ptf[4]
LU 0-3 RU 検査。 このバイトは、非標準の SNA データを送信するホスト・システムの次善策を提供するために使用されます。以下に説明する特定の問題が発生していない限り、 **追加なし** に設定する必要があります。

LU 0 から 3 の RU で CS Linux の通常の検査を使用するには、このパラメーターを **追加なし**に設定します。

LU 0 から 3 の RU に関する特定のチェックをリラックスするには、次の値を指定します。

追加の許可 _RQE
SNA プロトコルは BB ! という状態です。LU 0 から 3 の PLU-SLU セッションでの EB RU は、 RQD でなければなりません。 複数のホストが RQE BB を送信して EB CD - CS Linux が常に許容するプロトコル違反です。 この値が設定されている場合、 CS Linux は RQE BB を許容します。エブ !CD EC RU も同様に使用できます。

SEND_D_ACTLU_POWER_ON (電源オン)
アプリケーションが LU 0-3 LU を使用している場合 (例えば、その LU に対して RUI_INIT が受け取られ、 ACTLU が受信された場合) は、このオプションは CS Linux が、サブベクトルの電源を入れた +ve RSP ACTLU で応答する必要があることを示します。 このフラグを指定しないと、 CS Linux は、このサブベクトルなしで ACTLU RSP を送信し、後続の NOTIFY メッセージは電源オン状態を示します。

cp_create_parms.ptf[5]
受信した接続のセキュリティー検査。

ローカル呼び出し可能 TP が会話セキュリティーを必要としない、または定義されていないため、会話セキュリティーを必要としないデフォルトの場合、呼び出し元 TP は、それにアクセスするためにユーザー ID とパスワードを送信する必要はありません。 呼び出し元 TP がこれらのパラメーターを提供し、 CS Linux が受信する Attach メッセージにそれらのパラメーターが含まれている場合、 CS Linux は、呼び出し可能 TP が会話セキュリティーを必要としない場合でも、通常はパラメーターを検査し (また、無効な場合は Attach を拒否します) ます。 このパラメーターを使用すると、検査を無効にできます。 このバイトは、以下のいずれかの値に設定します。

追加なし
呼び出し可能 TP (通常の処理) のセキュリティー要件に関係なく、受信した Attach に含まれている場合は、常にセキュリティー・パラメーターを確認します。

AP_LIMIT_TP_SECURITY
呼び出し可能 TP がそれを必要としない場合は、受信した Attach 上のセキュリティー・パラメーターを検査しません。

cp_create_parms.ptf[6]
HPR の RTP オプション。

通常の RTP 処理を使用するには、 CS Linux がリモート・システムの機能に応じて使用可能な最良の RTP メカニズムを使用するように、このパラメーターを **追加なし**に設定します。

RTP 操作をカスタマイズするには、次のいずれかの値を指定します。

追加進行中の ARB
この値が設定されている場合、 CS Linux は、標準モードおよび応答モード ARB アルゴリズムのサポートを公示しますが、プログレッシブ・モード・アルゴリズムはサポートしません。

cp_create_parms.ptf[7]
DACTLU での DLUR アンバインド。 CS Linux は、 DLUR を使用するセッションのためにホストから DACTLU を受信したときに、通常は PLU-SLU セッションを終了しません。 このバイトは、以下のいずれかの値に設定します。

追加なし
通常処理を使用します。

付加的な DLACT_ON_DACTLU
DLUR を使用するセッションで DACTLU を受信した場合は、 PLU-SLU セッションを終了します。

cp_create_parms.ptf[8]
REQACTPU の PU 名を抑止します。 CS Linux は、 DLUR PU の活動化時に、 REQACTPU メッセージ内の PU 名を識別します。 このバイトは、以下のいずれかの値に設定します。

追加なし
通常処理を使用します。

AP_SUPPRESS_PU_NAME_ON_REQACTPU
DLUR PU の活動化時に PU 名を抑止します。

BIND_CNOS_ON_BIND_RSP の追加入力
APPC セッションの活動化中に、パートナー・システム上の一時的な条件が原因で CNOS セッションの活動化が失敗する可能性があります。 特定のセンス・コードによって示される条件は、常に (タイマーを使用して) 再試行されます。 このフラグを設定すると、 CS Linux は常に失敗した CNOS セッションのアクティブ化を再試行します。

cp_create_parms.ptf[9]
RUI ブラケット・レース・オプション、接続ネットワークの限定リソース・オーバーライド・オプション、および TCP/IP 情報制御ベクトル・オプション。

RUI アプリケーションがブラケット・プロトコルを使用していて、ホストが既に RUI アプリケーションの送信後に BB (開始ブラケット) を送信する場合、 CS Linux は通常、センス・データ 0813 ではこれをリジェクトし、アプリケーションには渡しません。 このバイトは、以下のいずれかの値に設定します。

追加なし
通常処理を使用します。

パス・パスの追加
BBR を RUI アプリケーションに渡します。 アプリケーションは、センス・データが 0813 または 0814 のいずれかの否定応答を送信する必要があります。

接続ネットワークを使用する CS Linux 内のリンクは、通常は限定リソースです。 これをオーバーライドするには、上記の値を以下の値と結合します。

CN_OVERRIDE_LIM_RES を指定しています
各接続ネットワーク・リンクに関連付けられたポートの 暗黙の制限リソース パラメーターを使用して、それが限定されたリソースであるかどうかを判別します。

CS Linux は通常、 TN3270 または LUA セッションのために、ホストへの NOTIFY 要求で TCP/IP 情報制御ベクトル (0x64) を組み込みます。 このベクトルには、ホスト・コンソールに表示できる情報、またはホストで使用される情報 (例えば、請求書など) が含まれます。クライアントが使用する TCP/IP アドレスとポート番号、およびクライアント・アドレスに対応する IP 名が含まれます。 TN3270 の場合、 TN3270 サーバーは通常、ドメイン・ネーム・サーバー (DNS) ルックアップを実行して、クライアント IP 名を判別します。

クライアント・アドレスが IPv6 アドレスであるが、ホストが IPv6 アドレスを解釈できないバックレベル・バージョンの VTAM を実行している場合は、ホスト・コンソールでクライアント・アドレスが正しく表示されない可能性があります。

以下のフラグを使用すると、この動作をオーバーライドできます。 これを行うには、上記の値を以下の値のいずれかに結合します。

付加 TCPIP_ベクトル
TN3270 または LUA のいずれかのホストに、 NOTIFY 要求に TCP/IP 情報制御ベクトル (0x64) を含めないでください。

ホストが、この制御ベクトルをサポートしていない古いバージョンの VTAM を実行している場合は、この値を使用してください。

名前のノード名 （ _TCPIP_NAME ）
DNS ルックアップを実行せず、 CV64 制御ベクトルをクライアントの IP アドレスで送信しますが、 IP 名は送信しません。

この値は TN3270 にのみ適用されます。 LUA クライアントには DNS ルックアップは必要ありません。 この値は、 DNS 環境が低速である場合、またはクライアントが DNS データに含まれていないことが分かっている場合 (例えば、 DDNS のない DHCP クライアントである 場合など) を使用します。

cp_create_parms.ptf[10]
FMH-5 接続メッセージの論理作業単位 ID (LUWID) を抑制します。 CS Linux には通常、 APPC 会話を開始するために送信する FMH-5 接続メッセージに、 LUWID が含まれます。 このバイトは、以下のいずれかの値に設定します。

追加なし
通常処理を使用します。

追加されたサイズの値を追加する
FMH-5 接続には LUWID を指定しないでください。このフィールドのフィールド 長をゼロとして指定してください。

cp_create_parms.ptf[11]
LU 管理オプション。

通常 LU 処理を使用するには、このパラメーターを **追加なし**に設定します。

LU 管理をカスタマイズするには、次の値のいずれかを指定します。

AP_DLUR_USE_REX_ペーシング
アップストリーム LU からの BIND がアダプティブ・ペーシングを無制限のペーシング・ウィンドウで要求する場合、 CS Linux は通常、ウィンドウ・サイズ 0 (ゼロ) を指定することによってこれを示します。 ダウンストリーム LU が適応ペーシングをサポートしていない場合は、このゼロ値を "ペーシングなし"と誤って解釈する可能性があるため、 CS Linux は、代わりにゼロ以外のペーシング・ウィンドウ・サイズを指定する必要があります。 このオプションを設定すると、 REX ステージ・ペーシング値が、ダウンストリーム LU に指定されたペーシング・ウィンドウ・サイズとして ACTLU から使用されます。

属性の追加が CLI_OVERWRITE_SYS_NAME
このオプションを設定すると、クライアント上で実行される APPC アプリケーションとそのアプリケーションが使用するプールされた LU との間の関連付けが維持されるため、パートナー・アプリケーションによって開始される後続の会話は正しいクライアントに経路指定されます。 クライア

ント・アプリケーションがプール内の LU にアクセスすると、 CS Linux は、 LU 上の システム名 パラメーターを、アプリケーションが実行されているクライアント・コンピューターのホスト名に変更します。 クライアントの管理の詳細については、「*IBM Communications Server for Linux* 管理ガイド上のデータ・センター・デプロイメント」を参照してください。

AP_OVERWRITE_INTERNAL_PU_PARMS
通常、 DLUR PU が定義されると、最初にその PU を削除することなく、その PU 上の構成パラメーターを変更する方法はありません (また、関連する LU)。 このフラグを設定すると、 CS Linux は、 snaadmin を使用して DLUR PU の新しい定義を受け入れ、そのノードが非アクティブであることも指定できます。 デフォルト以外のすべてのパラメーターを定義する必要があります (これは、 **ユーザー管理** -c コマンドと同等ではありません)。

cp_create_parms.cos_table_version
ノードによって使用される COS テーブルのバージョンを指定します。 このバイトは、以下のいずれかの値に設定します。

AP_VERSION_0_COS_TABLES
APPN アーキテクチャー解説書で最初に定義されている COS テーブルを使用してください。

アプリケーション・バージョン 1_COS_TABLES
ATM 上で HPR に対して最初に定義された COS テーブルを使用します。

cp_create_parms.send_term_self
PLU-SLU セッションをホストに終了するためのデフォルト方式を指定します。 指定した値は、ノード上のすべてのタイプ 0 から 3 の LU に使用されます。ただし、 LU 定義で別の値を指定することによってオーバーライドする場合を除きます。 次の値のいずれかを指定してください。

類人猿
CLOSE_PLU_SLU_SEC_RQ の受信時に TERM_SELF を送信します。

アプ・ノー
CLOSE_PLU_SLU_SEC_RQ の受信時に UNBIND を送信します。

***cp_create_parms.disable_branch_*認知**
このパラメーターは、 ノード・タイプ が AP_NETWORK_NODE の場合にのみ適用され、他のノード・タイプ用に予約済みです。

ローカル・ノードが次のいずれかの値を使用して、分岐認識をサポートするかどうかを指定します。 APPN オプション・セット 1120。

類人猿
ローカル・ノードはブランチ認識をサポートしていません。 このノードと提供される分岐ネットワーク・ノードの間の TG は、ネットワーク・トポロジーには表示されず、ローカル・ノード自体は分岐認識として報告されません。

アプ・ノー
ローカル・ノードは分岐認識をサポートします

cp_create_parms.cplu_syncpt_support
ノードの制御点 LU が同期点機能をサポートするかどうかを指定します。 このパラメーターは、 DEFINE_LOCAL_LU の 同期サポート パラメーターと同等ですが、ノードの制御点 LU にのみ適用されます (これには明示的な LU 定義はありません)。

このパラメーターを **類人猿** に設定するのは、標準の CS Linux 製品に加えて、同期点マネージャー (SPM) と会話保護リソース・マネージャー (C-PRM) がある場合のみにしてください。 可能な値は次のとおりです

類人猿
同期点がサポートされます。

アプ・ノー
同期点はサポートされていません。

cp_create_parms.cplu_attributes
ノードの制御点 LU に関する追加情報を識別します。 このパラメーターは、 DEFINE_LOCAL_LU の *lu_attributes* パラメーターと同等ですが、ノードの制御点 LU にのみ適用されます (これには明示的な LU 定義はありません)。

可能な値は次のとおりです

追加なし
追加情報はありません。

PWSUB を使用不可にする
制御点 LU のパスワード置換サポートを使用不可にします。 パスワード置換とは、平文として送信されるのではなく、ローカル LU とリモート LU の間でパスワードが暗号化されることを意味します。 CS Linux は、リモート・システムがパスワード置換をサポートしている場合、通常はパスワード

この値は、パスワード置換を正しく実装しないいくつかのリモート・システムとの通信のために、回避策として提供されます。 このオプションを使用する場合は、パスワードを平文で送信および受信する必要があることに注意する必要があります (セキュリティー・リスクを表す可能性があります)。 リモート・システムのパスワード置換の実装に問題がない限り、これを設定しないでください。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

target_handle
後続の verb で使用するための戻り値。

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_ISR_THRESHOLDS
The ISR threshold parameters were not valid (lower threshold above upper, or upper threshold above *max_isr_sessions*).

AP_INVALID_NODE_NAME
The *node_name* parameter contained a character that was not valid.

AP_INVALID_CP_NAME
The *cp_alias* or *fqcp_name* parameter contained a character that was not valid.

AP_INVALID_NODE_TYPE
The *node_type* parameter was not set to a valid value.

AP_PU_CONC_NOT_SUPPORTED
This version of CS Linux does not support the SNA gateway feature.

AP_DLUR_NOT_SUPPORTED
This version of CS Linux does not support the DLUR feature.

AP_INVALID_REG_WITH_NN
The *reg_with_nn* parameter was not set to a valid value.

AP_INVALID_COS_TABLE_VERSION
The *cos_table_version* parameter was not set to a valid value.

AP_INVALID_SEND_TERM_SELF
The *send_term_self* parameter was not set to a valid value.

AP_INVALID_DISABLE_BRANCH_AWRN
The *disable_branch_awareness* parameter was not set to a valid value.

AP_INVALID_DLUR_SUPPORT
The *dlur_support* parameter was not set to a valid value.

AP_INVALID_HPR_SUPPORT
The *hpr_support* parameter was not set to a valid value.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: state check

If the verb does not execute because of a state error, CS Linux returns the following parameters.

primary_rc
AP_STATE_CHECK

secondary_rc

AP_NODE_ALREADY_STARTED
The target node is active, so you cannot use this verb to modify its configuration. DEFINE_NODE can be issued only to an inactive node.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_STATE_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

DEFINE_PARTNER_LU

The DEFINE_PARTNER_LU verb 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 CS Linux will set up an implicit definition when the session to the partner LU is established; you should 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, note that you need to define a directory entry for the partner LU to allow CS Linux to access it. This can be done using either DEFINE_ADJACENT_LEN_NODE or DEFINE_DIRECTORY_ENTRY. 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 CS Linux can locate the LU dynamically.

VCB structure

```
typedef struct define_partner_lu
{
   AP_UINT16        opcode;              /* verb operation code             */
   unsigned char    reserv2;             /* reserved                        */
   unsigned char    format;              /* reserved                        */
   AP_UINT16        primary_rc;          /* primary return code             */
   AP_UINT32        secondary_rc;        /* secondary return code           */
   PLU_CHARS        plu_chars;           /* partner LU characteristics      */
} DEFINE_PARTNER_LU;
```

```
typedef struct plu_chars
{
   unsigned char    fqplu_name[17];      /* fully qualified partner LU name */
   unsigned char    plu_alias[8];        /* partner LU alias                */
   unsigned char    description[32];     /* resource description            */
   unsigned char    reserv2[16];         /* reserved                        */
```

```
    unsigned char    plu_un_name[8];       /* partner LU uninterpreted name   */
    unsigned char    preference;           /* reserved                        */
    AP_UINT16        max_mc_ll_send_size;  /* maximum MC send LL size         */
    unsigned char    conv_security_ver;    /* already-verified security       */
                                           /*  supported?                     */
    unsigned char    parallel_sess_supp;   /* parallel sessions supported?    */
    unsigned char    reserv3[8];           /* reserved                        */
} PLU_CHARS;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
保留中の PARTNER_LU

plu_chars.fqplu_name
パートナー LU の完全修飾 LU 名。 この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

この名前は、他のパートナー LU の完全修飾パートナー LU 名、またはローカル LU の LU 名と一致してはなりません。

plu_chars.plu_alias
パートナー LU の LU 別名。 これは 8 バイトの ASCII ストリングで、ローカルで表示可能な文字を使用し、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。

上記の *fqplu_name* パラメーターが、既存のパートナー LU の完全修飾名と一致する場合、このパラメーターは、既存の定義内のパートナー LU の別名と一致していなければなりません。 既存のパートナー LU のパートナー LU 別名を変更することはできません。また、同じ完全修飾名に対して複数の LU 別名をセットアップすることもできません。

plu_chars.description
パートナー LU を記述するヌルで終了するテキスト・ストリング (0 から 31 文字の後にヌル文字を続けたストリング) 。 このストリングは情報専用です。このストリングはノードの構成ファイルに保管され、 QUERY_PARTNER_LU verb および QUERY_PARTNER_LU_DEFINITION verb で戻されますが、 CS Linux ではそれ以外の使用は行われません。

plu_chars.plu_un_name
パートナー LU (リモート SSCP に定義されている LU の名前) の未解釈の名前。 この名前は、 8 バイトの EBCDIC 文字ストリングです。

デフォルトの未解釈名 (上記の *fqplu_name* パラメーターから取得したネットワーク名と同じ) を使用するには、このパラメーターを 8 桁の 2 進ゼロに設定します。 このパラメーターが関係するのは、パートナー LU がホスト上にあり、従属 LU 6.2 がそれにアクセスするために使用されている場合のみです。

plu_chars.max_mc_ll_send_size
パートナー LU でマップ式会話サービスが送信および受信できる論理レコードの最大サイズ。 1-32,767 の範囲内の数値を指定するか、または制限を指定しない場合にはゼロを指定します (この例では、最大は 32,767 です) 。

plu_chars.conv_security_ver
パートナー LU がローカル LUs; の代わりにユーザー ID の妥当性検査を行うことを許可されているかどうかを指定します。つまり、パートナー LU が Attach 要求で既に検査済みの標識を設定できるかどうかです。 可能な値は次のとおりです

類人猿
パートナー LU は、ユーザー ID を検証できます。

アプ・ノー
パートナー LU はユーザー ID を検証できません。

plu_chars.parallel_sess_supp
パートナー LU が並列セッションをサポートするかどうかを指定 可能な値は次のとおりです

類人猿
パートナー LU は並列セッションをサポートします。

アプ・ノー
パートナー LU は並列セッションをサポートしていません。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_DEF_PLU_INVALID_FQ_NAME
The *fqplu_name* parameter contained a character that was not valid.

AP_INVALID_UNINT_PLU_NAME
The *plu_un_name* parameter contained a character that was not valid.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: state check

If the verb does not execute because of a state error, CS Linux returns the following parameters.

primary_rc
AP_STATE_CHECK

secondary_rc
Possible values are:

AP_INVALID_FQ_LU_NAME
The *fqplu_name* parameter matched the name of an existing local LU.

AP_PLU_ALIAS_CANT_BE_CHANGED
The *plu_alias* parameter of an existing partner LU cannot be changed.

AP_PLU_ALIAS_ALREADY_USED
The *plu_alias* parameter is already used for an existing partner LU with a different LU name.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_STATE_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

DEFINE_PORT

DEFINE_PORT is used to define a new port or modify an existing one. Before issuing this verb, you must issue the DEFINE_DLC verb to 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.

If you are defining a port that will accept incoming calls, see “Incoming calls” on page 171.

VCB 構造体

```
typedef struct define_port
{
  AP_UINT16       opcode;                 /* verb operation code           */
  unsigned char   reserv2;                /* reserved                      */
  unsigned char   format;                 /* reserved                      */
  AP_UINT16       primary_rc;             /* primary return code           */
  AP_UINT32       secondary_rc;           /* secondary return code         */
  unsigned char   port_name[8];           /* name of port                  */
  PORT_DEF_DATA   def_data;               /* port defined data             */
} DEFINE_PORT;
```

```
typedef struct port_def_data
{
  unsigned char   description[32];         /* resource description          */
  unsigned char   initially_active;        /* is the port initially active?  */
  unsigned char   reserv2[15];             /* reserved                      */
  unsigned char   dlc_name[8];             /* DLC name associated with port  */
  unsigned char   port_type;               /* port type                     */
  unsigned char   port_attributes[4];      /* port attributes               */
  unsigned char   implicit_uplink_to_en;   /* implicit EN links up or down?  */
  unsigned char   implicit_appn_links_len; /* reserved                      */
  unsigned char   reserv3;                 /* reserved                      */
  AP_UINT32       port_number;             /* port number                   */
  AP_UINT16       max_rcv_btu_size;        /* max receive BTU size          */
  AP_UINT16       tot_link_act_lim;        /* total link activation limit    */
  AP_UINT16       inb_link_act_lim;        /* inbound link activation limit  */
  AP_UINT16       out_link_act_lim;        /* outbound link activation limit */
  unsigned char   ls_role;                 /* initial link station role      */
  unsigned char   retry_flags;             /* reserved                      */
  AP_UINT16       max_activation_attempts; /* reserved                      */
  AP_UINT16       activation_delay_timer;  /* reserved                      */
  unsigned char   mltg_pacing_algorithm;   /* reserved                      */
  unsigned char   implicit_tg_sharing_prohibited; /* reserved               */
  unsigned char   link_spec_data_format;   /* reserved                      */
  unsigned char   limit_enable;            /* reserved                      */
  unsigned char   reserv1[6];              /* reserved                      */
  unsigned char   implicit_dspu_template[8]; /* implicit dspu template      */
  AP_UINT16       implicit_ls_limit;       /* implicit ls limit             */
  unsigned char   reserv4;                 /* reserved                      */
  unsigned char   implicit_dspu_services;  /* implicit DSPU support         */
  AP_UINT16       implicit_deact_timer;    /* deact timer for implicit LSs   */
  AP_UINT16       act_xid_exchange_limit;  /* activation XID exchange limit  */
  AP_UINT16       nonact_xid_exchange_limit;     /* non-activation XID       */
                                                 /*  exchange limit          */
  unsigned char   ls_xmit_rcv_cap;         /* LS transmit-receive capability */
  unsigned char   max_ifrm_rcvd;           /* maximum number of I-frames that*/
                                           /*  can be received              */
  AP_UINT16       target_pacing_count;     /* Target pacing count           */
  AP_UINT16       max_send_btu_size;       /* Desired maximum send BTU size  */
  LINK_ADDRESS    dlc_data;                /* DLC data                      */
  LINK_ADDRESS    hpr_dlc_data;            /* reserved                      */
  unsigned char   implicit_cp_cp_sess_support;   /* implicit links allow     */
                                                 /*  CP-CP sessions          */
  unsigned char   implicit_limited_resource;     /* implicit links are       */
                                                 /*  limited resource        */
  unsigned char   implicit_hpr_support;    /* Is HPR supported?             */
  unsigned char   implicit_link_lvl_error; /* Send HPR traffic on implicit   */
                                           /*  links using link-level error  */
                                           /*  recovery?                     */
  unsigned char   retired1;                /*  reserved                     */
  TG_DEFINED_CHARS default_tg_chars;       /* default TG chars              */
  unsigned char   discovery_supported;     /* reserved                      */
  AP_UINT16       port_spec_data_len;      /* length of port specification   */
                                           /*  data                         */
  AP_UINT16       link_spec_data_len;      /* length of link specification   */
                                           /*  data                         */
} PORT_DEF_DATA;
```

```
typedef struct link_address
{
  unsigned char   format;                /* type of link address          */
  unsigned char   reserve1;              /* reserved                      */
  AP_UINT16       length;                /* length                        */
  unsigned char   address[135];          /* address                       */
} LINK_ADDRESS;
```

```
typedef struct tg_defined_chars
{
  unsigned char   effect_cap;            /* effective capacity            */
  unsigned char   reserve1[5];           /* reserved                      */
  unsigned char   connect_cost;          /* connection cost               */
  unsigned char   byte_cost;             /* byte cost                     */
  unsigned char   reserve2;              /* reserved                      */
  unsigned char   security;              /* security                      */
  unsigned char   prop_delay;            /* propagation delay             */
  unsigned char   modem_class;           /* reserved                      */
  unsigned char   user_def_parm_1;       /* user-defined parameter 1      */
  unsigned char   user_def_parm_2;       /* user-defined parameter 2      */
  unsigned char   user_def_parm_3;       /* user-defined parameter 3      */
} TG_DEFINED_CHARS;
```

SDLC の場合、ポート固有データもリンク固有データも含まれません。

QLLC のポート固有データ :

```
typedef struct vql_port_spec_data
{
  V0_MUX_INFO     mux_info;        /* streams config info                */
  unsigned char   driver_name[13]; /* reserved                           */
  unsigned char   cud_mode;        /* matching required on CUD           */
  AP_UINT16       cud_len;         /* length of Call User Data in octets */
  unsigned char   cud[128];        /* Call User Data                     */
  unsigned char   add_mode;        /* matching reqd on called address    */
  AP_UINT16       add_len;         /* length of called address to match  */
  AP_UINT32       xtras;           /* reserved                           */
  AP_UINT32       xtra_len;        /* reserved                           */
} VQL_PORT_SPEC_DATA;
```

QLLC のリンク固有データ :

```
typedef struct vql_ls_spec_data
{
   V0_MUX_INFO    mux_info;               /* streams config info           */
   AP_UINT16      reserve1;               /* reserved                      */
   AP_UINT16      reserve2;               /* reserved                      */
   unsigned char  vc_type;                /* Virtual Circuit type          */
   unsigned char  req_rev_charge;         /* reserved                      */
   unsigned char  loc_packet;             /* reserved                      */
   unsigned char  rem_packet;             /* reserved                      */
   unsigned char  loc_wsize;              /* reserved                      */
   unsigned char  rem_wsize;              /* reserved                      */
   AP_UINT16      fac_len;                /* X.25 facilities length        */
   unsigned char  fac[128];               /* X.25 facilities               */
   AP_UINT16      retry_limit;            /* reserved                      */
   AP_UINT16      retry_timeout;          /* reserved                      */
   AP_UINT16      idle_timeout;           /* reserved                      */
   AP_UINT16      pvc_id;                 /* PVC logical channel identifier */
   AP_UINT16      sn_id_len;              /* reserved                      */
   unsigned char  sn_id[4];               /* reserved                      */
   AP_UINT16      cud_len;                /* length of any call user data  */
                                          /*  to send                      */
   unsigned char  cud[128];               /* actual call user data         */
   AP_UINT32      xtras;                  /* reserved                      */
   AP_UINT32      xtra_len;               /* reserved                      */
   unsigned char  rx_thruput_class;       /* reserved                      */
   unsigned char  tx_thruput_class;       /* reserved                      */
   unsigned char  cugo;                   /* reserved                      */
   unsigned char  cug;                    /* reserved                      */
   AP_UINT16      cug_index;              /* reserved                      */
   AP_UINT16      nuid_length;            /* reserved                      */
   unsigned char  nuid_data[109];         /* reserved                      */
   unsigned char  reserve3[2];            /* reserved                      */
   unsigned char  rpoa_count;             /* reserved                      */
   AP_UINT16      rpoa_ids[30];           /* reserved                      */
} VQL_LS_SPEC_DATA;
```

トークンリング、イーサネットのポート固有データ

```
typedef struct llc_port_spec_data
{
  V0_MUX_INFO        mux_info;              /* Streams config info           */
  PROM_MODE_DATA     prom_data;             /* reserved                      */
  LLC_SAP_SPEC_DATA  sap_spec_data;         /* LLC2 timeouts and thresholds  */
  unsigned char      adapter_id[8];         /* adapter ID                    */
  unsigned char      adapt_spec_data[16];   /* reserved                      */
  AP_UINT32          max_rcv_pool_kb;       /* max size of receive buf pool  */
  unsigned char      throttle_back_pcent;   /* throttle back threshold       */
  unsigned char      pad[3];                /* reserved                      */
} LLC_PORT_SPEC_DATA;
```

```
typedef struct prom_mode_data
{
  AP_UINT16       port_spec_data_size;      /* reserved                      */
  unsigned char   promiscuous;              /* reserved                      */
  unsigned char   dlsw_flag;                /* reserved                      */
  AP_UINT16       vrn;                      /* reserved                      */
  AP_UINT16       bridge_num;               /* reserved                      */
} PROM_MODE_DATA;
```

```
typedef struct llc_sap_spec_data
{
  AP_UINT16       ack_timeout;              /* acknowlegment timeout in ms   */
  AP_UINT16       p_bit_timeout;            /* Poll response timeout in ms   */
  AP_UINT16       t2_timeout;               /* acknowledgment delay in ms    */
  AP_UINT16       rej_timeout;              /*REJ response timeout in seconds*/
  AP_UINT16       busy_state_timeout;       /* remote busy timeout in seconds*/
  AP_UINT16       idle_timeout;             /* idle RR interval in seconds   */
  AP_UINT16       max_retry;                /* retry limit for any response  */
  AP_UINT16       upward_cred_q_threshold;  /* reserved                      */
  AP_UINT16       window_inc_threshold;     /* window increment count for    */
                                            /*  dynamic window algorithm     */
  AP_UINT16       pad;                      /* reserved                      */
} LLC_SAP_SPEC_DATA;
```

トークンリング、イーサネットのリンク固有データ

```
typedef struct llc_link_spec_data
{
  V0_MUX_INFO     mux_info;              /* Streams config info             */
  AP_UINT16       reserve1;              /* reserved                        */
  AP_UINT16       reserve2;              /* reserved                        */
  AP_UINT16       length;                /* reserved                        */
  AP_UINT16       xid_timer;             /* XID timeout value in seconds    */
  AP_UINT16       xid_timer_retry;       /* XID retry limit                 */
  AP_UINT16       test_timer;            /* TEST timeout value in seconds   */
  AP_UINT16       test_timer_retry;      /* TEST retry limit                */
  AP_UINT16       ack_timeout;           /* acknowledgment timeout in ms    */
  AP_UINT16       p_bit_timeout;         /* POLL response timeout in ms     */
  AP_UINT16       t2_timeout;            /* acknowledgment delay in ms      */
  AP_UINT16       rej_timeout;           /* REJ response timeout in seconds */
  AP_UINT16       busy_state_timeout;    /* remote busy timeout in seconds  */
  AP_UINT16       idle_timeout;          /* idle RR interval in seconds     */
  AP_UINT16       max_retry;             /* retry limit for any response    */
} LLC_LINK_SPEC_DATA;
```

マルチパス・チャネル (MPC)用のポート固有データ、 CS Linux for IBM Z のみ:

```
typedef struct chnl_port_spec_data
{
  V0_MUX_INFO     mux_info;          /* streams information                */
  AP_UINT16       tx_buffers;        /* reserved                           */
  AP_UINT16       rx_buffers;        /* reserved                           */
  AP_UINT32       speed;             /* reserved                           */
  unsigned char   reserv1[32]        /* pad and future expansion           */
} CHNL_PORT_SPEC_DATA;
```

マルチパス・チャネル (MPC)のリンク固有データ :

```
typedef struct chnl_link_spec_data
{
   V0_MUX_INFO     mux_info;          /* streams information                */
```

```
   AP_UINT16       device_end;        /* BlkMux protocol flag                */
   unsigned char   escd_port;         /* reserved                            */
   unsigned char   cuadd;             /* reserved                            */
   unsigned char   local_name[8];     /* reserved                            */
   unsigned char   remote_name[8];    /* reserved                            */
   unsigned char   reserv1[32]        /* pad and future expansion            */
} CHNL_LINK_SPEC_DATA;
```

エンタープライズ・エクステンダー (HPR/IP)のポート固有データ：

```
typedef struct ipdlc_port_spec_data
{
   V0_MUX_INFO    mux_info;              /* streams information             */
   unsigned char  if_name[46];           /* Local interface id or IP address */
} IPDLC_PORT_SPEC_DATA;
```

Enterprise Extender (HPR/IP)のリンク固有データ：

```
typedef struct ipdlc_link_spec_data
{
   V0_MUX_INFO    mux_info;             /* streams information              */
   AP_UINT16      ack_timeout;          /* ACK timer for command frames     */
   AP_UINT16      max_retry;            /* Retry limit for command frames   */
   AP_UINT16      liveness_timeout;     /* Liveness timer                   */
   unsigned char  short_hold_mode;      /* Run in short-hold mode           */
   unsigned char  remote_hostname[255]; /* Name of remote host to contact   */
} IPDLC_LINK_SPEC_DATA;
```

すべての DLC タイプのデータ：

```
typedef struct v0_mux_info
{
   AP_UINT16         dlc_type;              /* DLC implementation type     */
   unsigned char     need_vrfy_fixup;       /* reserved                    */
   unsigned char     num_mux_ids;           /* reserved                    */
   AP_UINT32         card_type;             /* type of adapter card        */
   AP_UINT32         adapter_number;        /* DLC adapter number          */
   AP_UINT32         oem_data_length;       /* reserved                    */
   AP_INT32          mux_ids[5];            /* reserved                    */
} V0_MUX_INFO;
```

トークンリングまたはイーサネットの場合、 **リンク・アドレス** 構造体内の アドレス パラメーターは、以下のように置き換えられます。

```
typedef struct tr_address
{
  unsigned char   mac_address[6];       /* reserved                        */
  unsigned char   lsap_address;         /* local SAP address               */
} TR_ADDRESS;
```

Enterprise Extender (HPR/IP) の場合、 **リンク・アドレス** 構造体内の アドレス パラメーターは、以下のように置き換えられます。

```
typedef struct ip_address_info
{
  unsigned char lsap;                     /* Local Service Access Point addr */
  unsigned char version;                  /* IPv4 or IPv6                    */
  unsigned char address[272];             /* IP Address or hostname          */

} IP_ADDRESS_INFO;
```

MPC の場合、完全な **リンク・アドレス** 構造体は予約されています。

Supplied parameters

The application supplies the following parameters:

opcode
AP_DEFINE_PORT

port_name
Name of port being defined. This is an 8-byte ASCII string, using any locally displayable characters, padded on the right with spaces if the name is shorter than 8 bytes.

def_data.description
A null-terminated text string (0-31 characters followed by a null character) describing the port. This string is for information only; it is stored in the node's configuration file and returned on the QUERY_PORT verb, but CS Linux does not make any other use of it.

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

AP_YES
The port is automatically started when the node is started.

AP_NO
The port is automatically started only if an LS that uses it is defined to be initially active; otherwise it must be started manually.

def_data.dlc_name
Name of associated DLC. This is an 8-byte ASCII string, using any locally displayable characters, padded on the right with spaces if the name is shorter than 8 bytes. The specified DLC must have already been defined by a DEFINE_DLC verb.

def_data.port_type
Type of line used by the port.

For SDLC, the following values are allowed:

AP_PORT_SWITCHED
Switched line.

AP_PORT_NONSWITCHED
Nonswitched line.

For QLLC, this parameter must be set to AP_PORT_SWITCHED.

For Token Ring / Ethernet , this parameter must be set to AP_PORT_SATF (shared access transport facility).

For Enterprise Extender (HPR/IP), this parameter must be set to AP_PORT_SATF (shared access transport facility).

For MPC, this parameter must be set to AP_PORT_SWITCHED.

def_data.port_attributes
This is a one-bit parameter that may take the following values:

AP_NO
Incoming calls are resolved by CP name.

AP_RESOLVE_BY_LINK_ADDRESS
This specifies that an attempt is made to resolve incoming calls by using the link address on CONNECT_IN before using the CP name (or node ID) carried on the received XID3 to resolve them. This bit is ignored unless the *port_type* parameter is set to AP_PORT_SWITCHED.

def_data.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. For an MPC port, this parameter is reserved because implicit links are not supported.

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:

AP_YES
Implicit links to an End Node are uplinks.

AP_NO
Implicit links to an End Node are downlinks.

def_data.port_number
The number of the port.

For Enterprise Extender (HPR/IP), this parameter is reserved.

For MPC, this is the number corresponding to the MultiPath Channel device: for example, port 0 is `/dev/mpc0` and port 1 is `/dev/mpc1`.

def_data.max_rcv_btu_size
Maximum BTU size that can be received. 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 (265-4105 for SDLC).

def_data.tot_link_act_lim
Total link activation limit (the maximum number of links that can be active at any time using this port).

For an SDLC port with *port_type* set to `AP_NONSWITCHED` and *ls_role* set to `AP_LS_PRI`or `AP_LS_SEC`, the range is 1-256 (a value greater than 1 defines a multi-drop primary link or a multi-PU secondary link). For all other SDLC ports, this parameter must be set to 1.

For an MPC link, this parameter must be set to 1.

def_data.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.

For an SDLC port with *port_type* set to `AP_NONSWITCHED`, this parameter must be zero. If *port_type* is set to `AP_SWITCHED`, then the port must be defined to accept either incoming calls (*inb_link_act_lim = 1* and *out_link_act_lim = 0*) or outgoing calls (*inb_link_act_lim = 0* and *out_link_act_lim = 1*).

def_data.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.

For an SDLC port with *port_type* set to `AP_NONSWITCHED`, this parameter must be equal to *tot_link_act_lim*. If *port_type* is set to `AP_SWITCHED`, then the port must be defined to accept either incoming calls (*inb_link_act_lim = 1* and *out_link_act_lim = 0*) or outgoing calls (*inb_link_act_lim = 0* and *out_link_act_lim = 1*).

def_data.ls_role
Link station role.

For SDLC or QLLC, the following values are allowed:

AP_LS_PRI
Primary

AP_LS_SEC
Secondary

AP_LS_NEG
Negotiable

For Token Ring / Ethernet/ Enterprise Extender (HPR/IP), this must be set to `AP_LS_NEG`.

For MPC, this must be set to `AP_LS_NEG`.

def_data.implicit_dspu_template
For Enterprise Extender (HPR/IP), MPC, this parameter is reserved.

Specifies the DSPU template, defined on the DEFINE_DSPU_TEMPLATE verb. 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 AP_PU_CONCENTRATION, the *implicit_dspu_template* parameter is reserved.

def_data.implicit_ls_limit
Specifies the maximum number of implicit link stations that 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 AP_NO_IMPLICIT_LINKS indicates that no implicit links are allowed.

def_data.implicit_dspu_services
For Enterprise Extender (HPR/IP), MPC, this parameter is reserved.

Specifies the services that the local node will provide to the downstream PU across implicit links activated on this port. Possible values are:

AP_DLUR
Local node will provide DLUR services for the downstream PU (using the default DLUS configured through the DEFINE_DLUR_DEFAULTS verb).

AP_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*.

AP_NONE
Local node will provide no services for this downstream PU.

def_data.implicit_deact_timer
If *implicit_hpr_support* is set to AP_YES and *implicit_limited_resource* is set to AP_NO_SESSIONS, an HPR-capable 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.

Implicit limited resource link deactivation timer (in seconds). If *implicit_limited_resource* is set to AP_INACTIVITY, an implicit link using this port will be 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. Zero indicates no timeout (the link is not deactivated, as though *implicit_limited_resource* were set to AP_NO). This parameter is reserved if *implicit_limited_resource* is set to any value other than AP_INACTIVITY.

def_data.act_xid_exchange_limit
Activation XID exchange limit.

def_data.nonact_xid_exchange_limit
Non-activation XID exchange limit.

def_data.ls_xmit_rcv_cap
This parameter is reserved.

Specifies the link station transmit/receive capability. Possible values are:

AP_LS_TWS
Two-way simultaneous

AP_LS_TWA
Two-way alternating

For Enterprise Extender (HPR/IP), this parameter must be set to AP_LS_TWS.

def_data.max_ifrm_rcvd
Maximum number of I-frames that can be received by the local link stations before an acknowledgment is sent. Range: 1-127.

def_data.target_pacing_count
Numeric value between 1 and 32,767 inclusive indicating the desired pacing window size. (The current version of CS Linux does not make use of this value.)

def_data.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 (265-4105 for SDLC).

def_data.dlc_data.format
The type of link address specified for this port. Possible values:

AP_IP_ADDRESS_INFO
IP address. Specify this value for an Enterprise Extender (HPR/IP) port.

AP_UNSPECIFIED
Unspecified address format. Specify this value for any port type other than Enterprise Extender (HPR/IP).

def_data.dlc_data.length
For MPC, this parameter is reserved.

Length of the port address (in the following parameter).

def_data.dlc_data.address
Port address.

For MPC, this parameter is reserved.

For SDLC, this is a 1-byte address. If *ls_role* is set to AP_LS_SEC, or if *ls_role* is set to AP_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 local station is primary, or if the port is used only for outgoing calls, this parameter is reserved.

For QLLC, this is a string of 1-14 bytes, specifying the local X.25 DTE address of the port. This address must match the address configured in your X.25 driver for this network.

Note : If no address is specified on a QLLC port, an outgoing call request generated by CS Linux will not contain the X.25 calling address. Some hosts require this address as a security measure on incoming calls, and may not accept the connection without it.

def_data.dlc_data.tr_address.lsap_address
For Token Ring or Ethernet: Local SAP address of the port. Specify a multiple of 0x04 in the range 0x04-0xEC.

(The first parameter in the address structure normally contains the MAC address, but this value is used only on the LS and is reserved on the port.)

def_data.dlc_data.ip_address_info.lsap
For Enterprise Extender: Local SAP address of the port. Specify a multiple of 0x04 in the range 0x04-0xEC. The usual value is 0x04, but VTAM may use 0x08 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.

def_data.dlc_data.ip_address_info.version
For Enterprise Extender: Specifies whether the following field represents an IPv4 or IPv6 address. All link stations that use the port must use the same type of address. You cannot change this parameter if one or more link stations already use this port. Possible values:

IP_VERSION_4_HOSTNAME
The *address* field specifies an IPv4 address, or a hostname or alias that resolves to an IPv4 address.

IP_VERSION_6_HOSTNAME
The *address* field specifies an IPv6 address, or a hostname or alias that resolves to an IPv6 address.

def_data.dlc_data.ip_address_info.address
For Enterprise Extender: IP address of the port. 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`).

def_data.implicit_cp_cp_sess_support
Specifies whether CP-CP sessions are permitted for implicit link stations using this port. Possible values are:

AP_YES
CP-CP sessions are permitted for implicit LSs.

AP_NO
CP-CP sessions are not permitted for implicit LSs.

def_data.implicit_limited_resource
Specifies whether implicit link stations off this port should be defined as limited resources. Possible values are:

AP_NO
Implicit links are not limited resources, and will not be deactivated automatically.

AP_NO_SESSIONS
Implicit links are limited resources, and will be deactivated automatically when no active sessions are using them.

AP_INACTIVITY
Implicit links are limited resources, and will be deactivated automatically when no active sessions are using them or when no data has flowed for the time period specified by the *implicit_deact_timer* field.

- 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* field of the DEFINE_NODE verb.

def_data.implicit_hpr_support
Specifies whether High Performance Routing (HPR) is supported on implicit links. Possible values are:

AP_YES
HPR is supported on implicit links.

AP_NO
HPR is not supported on implicit links.

For Enterprise Extender (HPR/IP), this parameter must be set to AP_YES.

def_data.implicit_link_lvl_error
For SDLC, Enterprise Extender (HPR/IP)MPC, this parameter is reserved.

Specifies whether HPR traffic should be sent on implicit links using link-level error recovery (AP_YES or AP_NO). The parameter is reserved if *implicit_hpr_support* is set to AP_NO.

def_data.default_tg_chars
Default TG characteristics. These are used for implicit link stations using this port, and as the default TG characteristics for defined link stations that do not have TG characteristics explicitly defined. The TG characteristics parameters are ignored if the LS is to a downstream PU.

For details of these parameters, see “DEFINE_LS” on page 104.

def_data.port_spec_data_len
Length of port-specific data. The data should be concatenated to the basic VCB structure.

Port-specific data is not used for SDLC; set this parameter to zero.

def_data.link_spec_data_len
Length of link-specific data. The link-specific data should be concatenated immediately following the port-specific data.

For details of these parameters, see “DEFINE_LS” on page 104; the values specified on DEFINE_PORT are used as defaults for processing incoming calls (when the LS name is not initially known).

Port-specific data for QLLC:

mux_info.dlc_type
Type of the DLC. Set this to AP_IMPL_NLI_QLLC.

cud_mode
Specifies the type of matching required between the Call User Data (CUD) supplied on an incoming call and the *cud* parameter below. Possible values are:

VQL_DONTCARE
CUD on incoming calls is not checked.

VQL_IDENTITY
The received CUD must match the string specified in the *cud* parameter.

VQL_STARTSWITH
The initial bytes (up to *cud_len*) of the received CUD must match the string specified in the *cud* parameter; any bytes following *cud_len* are not checked.

cud_len
Specifies the length of the Call User Data (in the *cud* parameter below).

cud
Call user data to be used for verifying incoming calls. If *cud_mode* above is set to VQL_IDENTITY or VQL_STARTSWITH, incoming calls are accepted only if they specify a CUD string that matches the value defined in this parameter. If *cud_mode* is set to VQL_DONTCARE, this 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 above. Possible values are:

VQL_DONTCARE
The address on incoming calls is not checked.

VQL_IDENTITY
The received address must match the string specified in the *address* parameter.

VQL_STARTSWITH
The initial bytes (up to *add_len*) of the received address must match the string specified in the *address* parameter; any bytes following *add_len* are not checked.

If the *address* parameter is set to a null string, this parameter must be set to VQL_DONTCARE.

add_len
If *add_mode* is set to VQL_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.

Port-specific data for Token Ring or Ethernet:

mux_info.dlc_type
Type of the DLC.

Possible values are:

AP_IMPL_TR_SNAP_LLC2
Token Ring

AP_IMPL_ETHER_SNAP_LLC2
Ethernet

sap_spec_data.ack_timeout
Timeout in milliseconds within which CS Linux expects a response to any I-frames sent to the adjacent link station.

sap_spec_data.p_bit_timeout
Time in milliseconds that CS Linux waits for a response to a frame sent with the POLL bit set.

sap_spec_data.t2_timeout
Period in milliseconds that CS Linux will withhold a response to a received I-frame to allow further I-frames to be received and acknowledged with the same RR, thus reducing acknowledgment traffic. At the latest, CS Linux will send the acknowledgment after this period, but may send the acknowledgment before this period is over.

sap_spec_data.rej_timeout
Time in seconds during which CS Linux expects to receive a response to an REJ frame.

sap_spec_data.busy_state_timeout
Time in seconds that CS Linux waits for indication of clearance of a busy condition at the adjacent link station.

sap_spec_data.idle_timeout
RR keep-alive interval in seconds for an otherwise idle link.

sap_spec_data.max_retry
Maximum permitted number of retries when waiting for any response or busy state to clear.

sap_spec_data.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.

adapter_id
Adapter identifier for the MAC device used by the port. Set this to the name of the DLC used by the port, as defined by a DEFINE_DLC verb.

throttle_back_pcent
Usage level, as a percentage of the receive buffer pool, at which to withhold granting a new send window to a remote LS.

max_rcv_pool_kb
Maximum memory, in Kbits, that can be used by the receive buffer pool.

Port-specific data for multipath channel (MPC):

mux_info.dlc_type
Type of DLC. Set this to `AP_IMPL_MPC_GDLC`.

Port-specific data for Enterprise Extender (HPR/IP):

mux_info.dlc_type
Type of the DLC. Set this to `AP_IP`.

if_name
Identifier for the local network adapter card to be used for the IP link, if you have access to multiple IP networks. If you have access to only one IP network, you can leave this field set to binary zeros.

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

- An interface identifier (such as `eth0` or `en0`).

- 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`).

To determine the interface identifier, run the command `ipconfig - a` on the server where the card is installed. This lists the interface identifiers and their associated IP addresses.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

ポートフォリオ・ポート名
ポート名 パラメーターが無効でした。

ファイル名の変更ができない
指定された *dlc_name* が定義済みの DLC と一致しませんでした

ファイル・ポート・タイプの追加
port_type パラメーターが有効な値に設定されていませんでした。

ファイルのサイズが無効です
最大 *rcv_btu_size* パラメーターが有効な値に設定されていませんでした。

追加の追加のロール
ls_role パラメーターが有効な値に設定されていませんでした。

リンクのアクティブ化制限の制限
活動化限界パラメーターの 1 つが、有効な値に設定されていませんでした。

エージェントの追加の無効化
max_ifrm_rcvd パラメーターが有効な値に設定されていませんでした。

追加の追加のロール
ls_role パラメーターが有効な値に設定されていませんでした。

付加的なサービス・サービス
implicit_dspu_services パラメーターが有効な値に設定されていませんでした。

パブリッシュ / パブリッシュがサポートされていません
implicit_dspu_services パラメーターが予約値に設定されました。

アプリケーション・テンプレート名の変更
implicit_dspu_template パラメーターに指定された DSPU テンプレートが無効でした。

ファイルの追加バージョン
バージョン パラメーターが、 1 つ以上のリンク・ステーションで使用されている既存のポートで変更されました。 ポートにリンク・ステーションが関連付けられている場合は、このパラメーターを変更できません。

認識されていないホスト （ _R ）
リモート・ホスト名 パラメーターに指定されたストリングを、有効な IP アドレスに解決できませんでした。

ファイルの追加が禁止されています
予約済みパラメーターがゼロ以外の値に設定されました。

リンク・リンクの形式が無効です
予約済みパラメーターがゼロ以外の値に設定されました。

通知の使用可能性 （ UPLINK ）
暗黙指定 *_en* パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc
可能な値は次のとおりです

エクスポート・ ポートがアクティブ
指定されたポートは現在活動状態であるため、変更できません。

重複ファイル・ ポート番号の追加
指定された ポート番号 を持つポートは既に定義されています。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **状態検査の追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

Incoming calls

If you are configuring a port that will accept 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 be used for these calls, because CS Linux will define one dynamically 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 define an LS explicitly, 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, CS Linux checks the address specified on the call against the addresses specified for LSs 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 defined dynamically. To ensure that the explicit LS definition (including the required PU name) is used, ensure that the address defined for this LS matches the address that will be supplied by the host or the downstream computer on the incoming call. For Token Ring / Ethernet, both the MAC and SAP addresses must match in order to select the correct LS.

定義済み CF_ACCESS

DEFINE_RCF_ACCESS は、 CS Linux リモート・コマンド機能 (RCF) へのアクセスを指定します。これは、 UNIX コマンド機能 (UCF) コマンドの実行に使用されるユーザー ID 、およびサービス・ポイント・コマンド機能 (SPCF) を使用して、管理コマンドを発行できる制限事項です。 SPCF および UCF の詳細については、「 *IBM Communications Server for Linux* 管理ガイド上のデータ・センター・デプロイメント」を参照してください。この verb を使用して、 SPCF および UCF へのアクセスを許可するか、または SPCF の 1 つのみにアクセスすることができます。

この verb は、ドメイン構成ファイルに対して発行する必要があります。この verb を使用して、初めて RCF アクセスを指定したり、既存の定義を変更したりすることができます。 CS Linux は、ノードの始動時にこれらのパラメーターに対して動作します。ノードの実行中にこれらのパラメーターが変更されると、ノードが停止して再始動されるまで、そのノードが稼働しているサーバーでは変更は有効になりません。

VCB 構造体

```
typedef struct define_rcf_access
{
    AP_UINT16        opcode;                /* Verb operation code          */
    unsigned char    reserv2;               /* reserved                     */
    unsigned char    format;                /* reserved                     */
    AP_UINT16        primary_rc;            /* primary return code          */
    AP_UINT32        secondary_rc;          /* secondary return code        */
    unsigned char    ucf_username[32];      /* UCF username                 */
    AP_UINT32        spcf_permissions;      /* SPCF permissions             */
    unsigned char    reserv3[8];            /* Reserved                     */
} DEFINE_RCF_ACCESS;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
アプリケーション・アクセス CF_ACCESS

***ucf_* ユーザー名**
UCF ユーザーの Linux ユーザー名を指定します。 このパラメーターは、ヌル終了 ASCII ストリングです。 **根**という名前は指定しないでください。 CS Linux は、セキュリティー上の理由から、 UCF コマンドを root として実行することを許可しないからです。

すべての UCF コマンドは、このユーザーのユーザー ID を使用して実行されます。このユーザー ID は、デフォルト・シェル、デフォルト・グループ ID 、およびこのユーザーの Linux システムで定義されているアクセス許可を使用して実行されます。

UCF へのアクセスを禁止するには、このパラメーターをヌル・ストリングに設定します。

***spcf_*許可**
SPCF を使用してアクセスできる CS Linux verb のタイプを指定します。 これを **追加なし** に設定すると、 SPCF へのアクセスを禁止するか、以下の 1 つ以上の値 (論理 **それとも**を使用して結合される) にアクセスできます。

カタログを許可するローカル
QUERY_* verb は許可されています。

ローカルで許可さないローカル
DEFINE_*、 SET_*、 DELETE_*、 ADD_*、および REMOVE_* verb、および INIT_NODE も許可されています。

アプリケーション・アクション・ローカル (_R)
"アクション" の verb は許可されます。 START_*、 STOP_*、 ACTIVATE_*、 DEACTIVATE_*、および、 APING、 INITIALIZE_SESSION_LIMIT、 CHANGE_SESSION_LIMIT、および RESET_SESSION_LIMIT があります。

カタログを許可するリモート
QUERY_* verb は、ドメイン内の任意のノードに向けられることが許可されています。

許可を許可さない (リモート)
DEFINE_*、 SET_*、 DELETE_*、 ADD_*、 REMOVE_*、および INIT_NODE verb は、ドメイン内のどのノードでも送信することが許可されます。

追加アクションを実行 (リモート)
START_*、 STOP_*、 ACTIVATE_*、 DEACTIVATE_*、 APING、 INITIALIZE_SESSION_LIMIT 、および RESET_SESSION_LIMIT verb は、ドメイン内のどのノードでも送信することが許可されています。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

ユーザー・ルートの付加 （ _UCF_USER_CANNOT_BE_ROOT ）
*ucf_* ユーザー名 パラメーターが **根**という名前を指定しましたが、これは許可されませ

付加的なセキュリティー・セキュリティー
*spcf_*許可 パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

DEFINE_RTP_TUNING

DEFINE_RTP_TUNING specifies parameters to be used when setting up RTP connections. After you issue this verb, the parameters you specify will be used for all future RTP connections until you modify them by issuing a new DEFINE_RTP_TUNING verb.

VCB structure

```
typedef struct define_rtp_tuning
{
  AP_UINT16         opcode;               /* verb operation code             */
  unsigned char     reserv2;              /* reserved                        */
  unsigned char     format;               /* reserved                        */
  AP_UINT16         primary_rc;           /* primary return code             */
  AP_UINT32         secondary_rc;         /* secondary return code           */
  unsigned char     path_switch_attempts; /* number of path switch attempts  */
  unsigned char     short_req_retry_limit; /* short request timer retry limit */
  AP_UINT16         path_switch_times[4]; /* path switch times               */
  AP_UINT32         refifo_cap;           /* maximum for refifo timer        */
  AP_UINT32         srt_cap;              /* maximum for short request timer */
  AP_UINT16         path_switch_delay;    /* minimum delay before path switch*/
  unsigned char     reserved[78];         /* reserved                        */
} DEFINE_RTP_TUNING;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DEFINE_RTP_TUNING

path_switch_attempts
Number of path switch attempts to set on new RTP connections. Specify a value in the range 1-255. If you specify 0(zero), CS Linux uses the default value of 6.

short_req_retry_limit
Number of times a Status Request is sent before CS Linux determines that an RTP connection is disconnected and starts Path Switch processing. Specify a value in the range 1-255. If you specify 0(zero), CS Linux uses the default value of 6.

path_switch_times
Length of time in seconds for which CS Linux attempts to path switch a disconnected RTP connection. This parameter is specified as four separate time limits for each of the valid transmission priorities in

order: AP_LOW, AP_MEDIUM, AP_HIGH, and AP_NETWORK. Each of these must be in the range 1-65535. The value you specify for each transmission priority must not exceed the value for any lower transmission priority.

If you specify 0(zero) for any of these values, CS Linux uses the corresponding default value as follows:

- 480 seconds (8 minutes) for AP_LOW
- 240 seconds (4 minutes) for AP_MEDIUM
- 120 seconds (2 minutes) for AP_HIGH
- 60 seconds (1 minute) for 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. Setting a value of 0 (zero) means that the timer is not limited and can take any value calculated by the protocol.

The default value for this parameter is 4000 milliseconds, with a minimum value of 5 milliseconds. If you specify a value in the range 1-4, the value of 5 will be used.

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. Setting a value of 0 (zero) means that the timer is not limited and can take any value calculated by the protocol.

The default value for this parameter is 8000 milliseconds, with a minimum value of 5 milliseconds. If you specify a value in the range 1-4, the value of 5 will be used.

path_switch_delay
Minimum delay in seconds before a path switch occurs. Specifying a delay avoids unnecessary path switch attempts caused by transient delays in network traffic, in particular when there is no other route available.

Specify a value in the range 0-65535. The default value is zero, indicating that a path switch attempt can occur as soon as the protocol indicates it is required.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_PATH_SWITCH_TIMES
The *path_switch_times* parameter 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.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

セキュリティー・アクセス・リスト

DEFINE_SECURITY_ACCESS_LIST は、特定のローカル LU または呼び出し可能 TP にアクセスできるユーザーのリストを定義します。そのため、その LU または TP へのアクセスは、指定されたユーザーに制限されます。 また、ユーザー名を既存のセキュリティー・アクセス・リストに追加するために使用することもできます。 リスト内のユーザー名は、 DEFINE_USERID_PASSWORD verb を使用して定義されます。

特定のローカル LU または呼び出し可能 TP へのアクセスを制限するには、以下のことを行う必要があります。

1. LU または TP の各許可ユーザーが、 DEFINE_USERID_PASSWORD verb を使用して定義されていることを確認してください。
2. これらのすべてのユーザー ID を含むセキュリティー・アクセス・リストを定義するには、 DEFINE_SECURITY_ACCESS_LIST verb を使用します。
3. LU または TP を定義する DEFINE_LOCAL_LU verb または DEFINE_TP verb で、このセキュリティー・アクセス・リストの名前を指定します。

セキュリティー・アクセス・リストが定義されているローカル LU または呼び出し可能 TP に着信割り振り要求が到着した場合、呼び出し側アプリケーションは、会話セキュリティーが使用されることを指示し、ユーザー ID を指定する必要があります。 CS Linux は、標準会話セキュリティー検査 (DEFINE_USERID_PASSWORD verb を使用して指定されたユーザー ID に対して) のほかに、着信割り振り要求内のユーザー ID を、 LU または TP 用に定義されたセキュリティー・アクセス・リストに対して検査し、ユーザー ID が一致しない場合は会話を拒否します。 LU と TP の両方にセキュリティー・アクセス・リストが定義されている場合には、ユーザー ID は両方のリストになければなりません。

ローカル LU または呼び出し可能 TP にセキュリティー・アクセス・リストが定義されていないが、会話セキュリティーを必要とするように構成されている場合は、標準会話セキュリティー検査が引き続き適用されます。

VCB structure

The DEFINE_SECURITY_ACCESS_LIST verb contains a variable number of `security_user_data` structures; these define the user names to be added to the security access list. The user name structures are included at the end of the `def_data` structure; the number of these structures is specified by the *num_users* parameter.

```
typedef struct define_security_access_list
{
  AP_UINT16         opcode;               /* verb operation code             */
  unsigned char     reserv2;              /* reserved                        */
  unsigned char     format;               /* reserved                        */
  AP_UINT16         primary_rc;           /* primary return code             */
  AP_UINT32         secondary_rc;         /* secondary return code           */
  unsigned char     list_name[14];        /* name of this list               */
  unsigned char     reserv3[2];           /* reserved                        */
  SECURITY_LIST_DEF def_data;             /* security access list            */
} DEFINE_SECURITY_ACCESS_LIST;
```

```
typedef struct security_list_def
{
  unsigned char     description[32];      /* description                     */
  unsigned char     reserv3[16];          /* reserved                        */
  AP_UINT32         num_users;            /* number of users being added     */
  unsigned char     reserv2[16];          /* reserved                        */
} SECURITY_LIST_DEF;
```

```
typedef struct security_user_data
{
  AP_UINT16         sub_overlay_size;     /* reserved                        */
```

```
  unsigned char     user_name[10];        /* user name                        */
} SECURITY_USER_DATA;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DEFINE_SECURITY_ACCESS_LIST

list_name
Name of the security access list. This is an ASCII string, padded on the right with spaces.

If this name matches an existing security access list, the users defined by this verb are added to the list; otherwise a new list is created.

def_data.description
A null-terminated text string (0-31 characters followed by a null character) describing the security access list. This string is for information only; it is stored in the node's configuration file and returned on the QUERY_SECURITY_ACCESS_LIST verb, but CS Linux does not make any other use of it.

def_data.num_users
Number of user names being defined by this verb. Each user must be specified by a `security_user_data` structure following the `def_data` structure.

For each user name in the list, up to the number specified in *num_users*, a `security_user_data` structure is required with the following information:

user_name
Name of the user.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

ファイル・ リスト名の変更
リスト名 パラメーターに、無効な文字が含まれていました。

ユーザー名の変更
指定されたユーザー名のうち 1 つ以上が無効でした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

DEFINE_TN3270_ACCESS

DEFINE_TN3270_ACCESS defines TN3270 access details for a particular client (or default TN3270 access details for all clients) using the TN3270 Server feature of CS Linux. (To define access details for a client using TN Redirector, use DEFINE_TN_REDIRECT.)

Each verb specifies details for one or more sessions. Each session is uniquely identified by the client address and the server port number. The DEFINE_TN3270_ACCESS verb can be used to define a new client, to define new sessions for use by an existing client, or to modify the session parameters. (To delete sessions from an existing client, use DELETE_TN3270_ACCESS.)

VCB 構造体

DEFINE_TN3270_ACCESS verb には、変数番号の tn3270_session_def_data 構造が含まれています。これらの構造は、ユーザーのセッションを定義します。 セッション構造は、 **データの定義** 構造の終わりに組み込まれます。これらの構造体の数は、 *num_sessions* パラメーターによって指定されます。

```
typedef struct define_tn3270_access
{
   AP_UINT16           opcode;             /* verb operation code            */
   unsigned char       reserv2;            /* reserved                       */
   unsigned char       format;             /* reserved                       */
   AP_UINT16           primary_rc;         /* primary return code            */
   AP_UINT32           secondary_rc;       /* secondary return code          */
   AP_UINT16           default_record;     /* is this the DEFAULT record?    */
   unsigned char       client_address[256]; /* address of TN3270 user         */
   TN3270_ACCESS_DEF_DATA def_data;
} DEFINE_TN3270_ACCESS;
```

```
typedef struct tn3270_access_def_data
{
   unsigned char       description[32];    /* Description - null terminated  */
   unsigned char       reserv1[16];        /* reserved                       */
   AP_UINT16           address_format;     /* Format of client address       */
   AP_UINT32           num_sessions;       /* Number of sessions being added */
   unsigned char       reserv3[64];        /* reserved                       */
} TN3270_ACCESS_DEF_DATA;
```

```
typedef struct tn3270_session_def_data
{
   AP_UINT16           sub_overlay_size;   /* reserved                       */
   unsigned char       description[32];    /* Session description            */
   unsigned char       tn3270_support;     /* Level of TN3270 support        */
   unsigned char       allow_specific_lu;  /* Allow access to specific LUs   */
   unsigned char       printer_lu_name[8]; /* Generic printer LU/pool        */
                                           /*   accessed                     */
   unsigned char       reserv1[6];         /* reserved                       */
   AP_UINT16           port_number;        /* TCP/IP port used to access     */
                                           /*  server                        */
   unsigned char       lu_name[8];         /* Generic display LU/pool        */
                                           /*  accessed                      */
   unsigned char       session_type;       /* Unused in current version      */
   unsigned char       model_override;     /* Unused in current version      */
   unsigned char       ssl_enabled;        /* Is this an SSL session?        */
   unsigned char       security_level;     /* SSL encryption strength        */
   unsigned char       cert_key_label[80]; /* Certificate key label          */
   unsigned char       listen_local_address[46];
                                           /* Local addr client connects to  */
  unsigned char        allow_ssl_timeout_to_nonssl;
                                           /* Allow non-SSL clients on SSL?  */
  unsigned char     reserv3[17];
   AP_UINT32           reserv4;            /* reserved                       */
} TN3270_SESSION_DEF_DATA;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
追加のアクセス権 3270_R

default_record
この verb がデフォルト・レコードを定義するかどうかを指定します。このデフォルト・レコードは、TCP/IP アドレスによって明示的に識別されない TN3270 ユーザーが使用 TN3270 ユーザーが TN サーバー・ノードに接続しようとし、ユーザーの TCP/IP アドレスが構成内の DEFINE_TN3270_ACCESS レコードと一致しないが、デフォルト・レコードが定義されている場合は、このレコードからのパラメーターが使用されます。 可能な値は次のとおりです

類人猿
この verb はデフォルトレコードを定義します。 クライアント・アドレス パラメーターおよび 宛先フォーマット パラメーターは予約されています。

アプ・ノー
この verb は通常の TN3270 ユーザー・レコードを定義します

デフォルト・レコードは、 TN サーバーが実行されているコンピューターの TCP/IP アドレスを判別できるすべての TN3270 ユーザーに対して、 TN サーバー機能へのアクセスを提供します。 TN サーバーの使用を特定のユーザー・グループに制限するには、デフォルト・レコードを組み込まないか、または 3270 LU または LU プールを構成して使用できないように構成しておく必要があります。

ほとんどのユーザーのデフォルト・レコードをセットアップすることもできますが、 1 つ以上の TCP/IP アドレスを明示的に除外することもできます これを行うには、 TN サーバー・ユーザーとして除外されるアドレスを定義し、 3270 LU または LU プールを構成しないままにしておきます。

クライアント・アドレス
TN3270 プログラムが実行されているコンピューターの TCP/IP アドレス。 これはヌル終了 ASCII ストリングです。これは、以下のいずれかにすることができます。 宛先フォーマット パラメーターは、それが IP アドレスであるか、または名前であるかを示します。

- IPv4 小数点付き 10 進数アドレス (`193.1.11.100` など) 。
- IPv6 コロン 16 進アドレス (`2001:0db8:0000:0000:0000:0000:0000:0000:1428:57ab` や `2001:db8::1428:57ab` など) 。
- 名前 (`newbox.this.co.uk` など) 。
- 別名 (newbox など) 。

名前または別名を使用する場合は、以下の制約事項が適用されます。

- Linux システムでは、名前または別名を完全修飾名 (ローカル TCP/IP 構成を使用するか、ドメイン・ネーム・サーバーを使用して) に解決できる必要があります。
- それぞれの名前または別名は、固有の完全修飾名に拡張する必要があります。同じ TN サーバー・ノードのユーザーに対して 2 つの名前を構成してはなりません。これは、同じ完全修飾名に解決されることになります。
- 完全修飾名は大文字と小文字を区別しません。例えば、 `Newbox.THIS.CO.UK` は `newbox.this.co.uk` と同等です。

データの説明の説明
オプションのテキスト・ストリング (0 から 31 文字の後にヌル文字が続きます) 。 このストリングは情報専用です。このストリングは構成ファイルに保管され、 QUERY_TN3270_ACCESS_DEF verb の **アクセス定義に一致する照会** 構造体に戻されますが、 CS Linux ではその情報は使用されません。 これを使用して、ユーザーを区別するための追加情報を保管することができます。

データ .address_address_format のフォーマット
クライアント・アドレス パラメーターのフォーマットを指定します。 可能な値は次のとおりです

宛先 IP
IP アドレス (IPv4 または IPv6 のいずれか)

宛先数 FQN
別名または完全修飾名

def_data.num_sessions
この verb によって定義または変更されるセッションの数。 各 TN3270 ユーザーは、セッションごとに異なる TCP/IP ポートを使用することによって、複数のセッションを持つ同じ TN サーバー・ノード

にアクセスすることができます。 各セッションは、 tn3270_access_def_data 構造の後に tn3270_session_def_data 構造体によって指定する必要があります。

それぞれのセッションごとに、以下の情報を示す tn3270_session_data 構造が必要です。

記述
オプションのテキスト・ストリング (0 から 31 文字の後にヌル文字が続きます)。 このストリングは情報専用です。このストリングは構成ファイルに保管され、 QUERY_TN3270_ACCESS_DEF verb の**アクセス定義に一致する照会** 構造体に戻されますが、 CS Linux ではその情報は使用されません。

***tn3270_*サポート**
TN3270 サポートのレベルを指定します。 可能な値は次のとおりです

追加 3270
TN3270E プロトコルを使用不可に設定します。

値 3270E
TN3270E プロトコルが使用可能であることを指定します。

TN3270 および TN3287 プロトコルは常に使用可能です。

AS/400 TN3270 クライアントの場合、このパラメーターは **値** 3270E に設定する必要があります。

allow_specific_lu
特定の LU へのアクセスが許可されるかどうか 可能な値は次のとおりです

類人猿
特定 LU へのアクセスは許可されます

アプ・ノー
特定 LU へのアクセスは許可されません。

printer_lu_name
このセッションが総称プリンター LU を要求する接続に使用するプリンター LU または LU プールの名前。 これは、右側に EBCDIC スペースが埋め込まれたタイプ A の EBCDIC ストリングです。 これは、このノードで定義されている LU タイプ 0 から 3 の LU タイプの名前、またはこのノード上の LU を含む LU プールの名前と一致していなければなりません。

単一のプリンター LU が指定されている場合、このプリンター LU は DEFINE_TN3270_ASSOCIATION verb によってどのディスプレイ LU にも関連付けられてはなりません。 プリンター LU プールが指定されている場合は、プール内のどのプリンター LU もディスプレイ LU に関連付けられていません。 1 つの LU が汎用プリンター LU としてアクセスされ、関連プリンター LU としてアクセスできるようにすると、その LU がすでに使用中であるために、その LU が関連するプリンター LU として使用できなくなることがあります。 (これらの規則は NOF API によって適用されません。)

このフィールドは、特定のプリンター LU セッションには影響しません。

ポート番号
TN3270 プログラムが TN サーバー・ノードにアクセスするために使用するサーバー TCP/IP ポートの番号。 ポート番号が、この TN3270 ユーザーのセッションのいずれかに定義されている既存のポート番号と一致する場合は、そのセッションの情報が置き換えられます。それ以外の場合は、新しいセッションが追加されます。

2 つ以上のセッション構造が同じ ポート番号 を使用する (同じ クライアント・アドレス または異なる 1 つの) 場合、 *listen_local_address* パラメーターは、それらすべての中で指定するか、またはいずれも指定しないでください。一部のセッションでは指定することはできませんが、それ以外のセッションでは指定しないでください。

lu_name
このセッションが総称ディスプレイ LU を要求する接続に使用する LU または LU プールの名前。 これは、右側に EBCDIC スペースが埋め込まれたタイプ A の EBCDIC ストリングです。 これは、このノードに定義されているタイプ 0 から 3 のディスプレイ LU 、またはこのノード上の LU を含む LU プールの名前と一致する必要があります。

LU 名を指定すると、指定された TCP/IP アドレスを持つ TN3270 プログラムは、この TN サーバー・ノード上の指定されたサーバー・ポート番号に接続することによって、一度に 1 つのセッションしか使用できません。 LU プールを指定すると、プログラムは、複数の総称ディスプレイ LU セッション

(または、この TN サーバーを使用する総称ディスプレイ LU セッションにアクセスできるプログラムの複数のコピー) を使用することができます。このノード上のプールから使用可能な LU の数まで、このセッションを複数使用することができます。

このパラメーターは、特定のディスプレイ LU セッションには影響しません。

ssl_enabled
このセッションが Secure Sockets Layer (SSL) を使用してサーバーにアクセスするかどうかを示します。

このパラメーターは、サーバー上で SSL をサポートするために必要な追加のソフトウェアをインストールしていない場合には予約されます。 これは、 NOF verb QUERY_NODE_LIMITS を使用して、 *ssl_* サポート パラメーターの値を検査することによって確認できます。

可能な値は次のとおりです

アプ・ノー
このセッションは SSL を使用しません。

類人猿
このセッションでは SSL を使用

エイプ _WITH_WITH_CLI_AUTH
このセッションは SSL を使用し、 TN サーバーではクライアント 認証を使用する必要があります。クライアントは、有効な証明書 (TN サーバーを使用する許可を与えられた有効なクライアントとして識別する情報) を送信する必要があります。

証明書が有効であることを確認するだけでなく、 TN サーバーは、外部 LDAP サーバー上の証明書失効リストに対して証明書を検査して、ユーザーの許可が取り消されていないことを確認する必要がある場合があります。 この場合は、 DEFINE_TN3270_SSL_LDAP を使用して、このサーバーへのアクセス方法を指定する必要があります。 ユーザーが TN3270 高速ログオン機能の使用を許可されている場合は、 DEFINE_TN3270_EXPRESS_LOGON を使用して、この機能をセットアップする必要もあります。

注 :

1. このセッションの ポート番号 パラメーターが、 Telnet デーモンの TCP/IP ポートを使用することを示している場合は、このセッションでは SSL を使用しないでください。 Telnet デーモンの TCP/IP ポートを使用するセッションで SSL を使用する場合、 Telnet クライアントは **テルネット** は、ノードがアクティブになっているときに CS Linux コンピューターにアクセスします。 を使用できません。
2. 同じポートを使用する多数のクライアントがあり、それらを非 SSL から SSL 構成にマイグレーションしている場合は、マイグレーションの進行中に、同じポートで SSL 接続と非 SSL 接続の両方を受け入れるように構成をセットアップすることができます。 以下の 許可されていない、 *to_nonssl* を指定して パラメーターを参照してください。

セキュリティー・レベル
このセッションに必要な SSL セキュリティー・レベルを示します。 セッションは、クライアントとサーバーの両方がサポートできる最高のセキュリティー・レベルを使用します。要求されたセキュリティーまたはそれ以上のレベルのセキュリティーをクライアントがサポートできない 場合、セッションは開始されません。

ssl_enabled パラメーターが **アプ・ノー**に設定されている場合、このパラメーターは予約されています。

可能な値は次のとおりです

最小の認証認証 (分)
証明書を交換する必要があります。暗号化は必須ではありません (ただし、クライアントが要求する場合は使用できます)

証明書認証のみを追加
証明書は交換しなければなりませんが、暗号化は使用されませ このオプションは通常、クライアントがセキュア・イントラネットを介して接続している場合の暗号化のオーバーヘッドを回避するために使用されます。

追加 40_BIT_MIN
40 ビット以上の暗号化を使用します。

最小値 56_BIT_MIN
56 ビット以上の暗号化を使用します。

最小値 128_BIT_MIN
128 ビット以上の暗号化を使用します。

最小 168_BIT_MIN
168 ビット以上の暗号化を使用します。

最小値の 256_BIT_MIN
256 ビット以上の暗号化を使用します。

注 : 暗号化を使用するには、 CS Linux とともにインストールする追加のソフトウェアが必要です。詳しくは、 *Linux* スタートアップ・ガイドでのデータ・センター・デプロイメントのための *IBM Communications Server* を参照して ご使用のロケーションによっては、リストされた暗号化レベルをすべて使用できない場合があります。これらの暗号化レベルをサポートするために必要なソフトウェアが国内で使用できないためです

cert_key_label
このセッションで SSL と使用するための証明書と鍵ペアを識別するラベル。 これは、 SSL 鍵リング・データベースのセットアップ時に指定されたラベルと一致する必要があります。詳しくは、 *Linux* スタートアップ・ガイドでのデータ・センター・デプロイメントのための *IBM Communications Server* を参照してください。

このラベルは、ヌル終了の ASCII 文字ストリングです。 SSL 鍵リング・データベースのセットアップ時に指定されたデフォルトの SSL 証明書および鍵ペアを使用するには、このパラメーターをヌル・ストリングに設定します。

listen_local_address
TN3270 クライアントが接続するローカル TN サーバー・コンピューター上のアドレス。

- TN3270 クライアントがローカル・アドレス上で接続できるようにする場合、または TN サーバー上に有効なローカル・アドレスが 1 つしかない場合は、このパラメーターをすべて 2 進ゼロに設定します。 この場合、このパラメーターと同じ ポート番号 を使用するすべての `tn3270_session_data` 構造体 (同じ クライアント・アドレス または別のもの) にも、このパラメーターがすべて 2 進ゼロに設定されている必要があります。
- TN3270 クライアントを特定のローカル・アドレスに制限する必要がある場合は、それをヌル終了 ASCII ストリングとして指定します。 このアドレスは、以下のいずれかになります。
  - IPv4 小数点付き 10 進数アドレス (`193.1.11.100` など) 。
  - IPv6 コロン 16 進アドレス (`2001:0db8:0000:0000:0000:0000:0000:1428:57ab` や `2001:db8::1428:57ab` など) 。

  アドレスを指定する場合、このアドレスと同じ ポート番号 を使用するすべての `tn3270_session_data` 構造体 (同じ クライアント・アドレス または別のもの) にも、このパラメーターに値が指定されている必要があります。ただし、アドレスは、すべてのセッションで同じではありません。

注 : 1 つ以上のセッションに対してローカル・アドレスを指定すると、このクライアント・レコードは Motif 管理プログラムに表示されないため、そのプログラムを使用してそのプログラムを表示または管理することはできません。 コマンド行管理プログラムを使用して、それを引き続き管理することができます `snaadmin`、または NOF アプリケーション。

許可されていない、*to_nonssl* を指定して
ssl_enabled が **アプ・ノー**に設定されている場合、このパラメーターは適用されません SSL を使用するように構成されていても、非 SSL TN3270 クライアントがこのセッション ・レコードを使用してサーバーにアクセスできるかどうかを示します。 可能な値は次のとおりです

類人猿
SSL を使用しない TN3270 クライアントは、サーバーにアクセスできます。 サーバーが SSL ネゴシエーションを開始するのを待っている間に、始動時に 5 秒間の遅延が発生します。これ以降、サーバーは、クライアントが SSL を使用しておらず、通常の TN3270 通信に戻ると想定します。

アプ・ノー
SSL を使用する TN3270 クライアントのみがサーバーにアクセスできます。

注 : このオプションは、マイグレーション目的で提供されます。同じポートを使用する多数のクライアントがあり、それらのクライアントを非 SSL から SSL 構成にマイグレーションする場合、マイグレーションの進行中に、同じポートで SSL 接続と非 SSL 接続の両方を受け入れるように構成をセットアップできます。

非 SSL クライアントが SSL リソースを使用できるようにすることは、機密漏れの可能性があるため、このオプションは長期使用を目的としたものではありません。 このパラメーターを **類人猿** に設定するのは、マイグレーションの進行中の短期間のみで、マイグレーションの完了時には **アプ・ノー** に設定する必要があります。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_UNKNOWN_CLIENT_ADDRESS
The specified name or alias could not be mapped to a fully qualified name.

AP_CLIENT_ADDRESS_CLASH
The fully qualified name, resolved from the *client_address* parameter, clashes with one that has already been defined.

AP_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.

AP_TCPIP_PORT_IN_USE
The TCP/IP port number cannot be used by TN server because it is already in use by a different program.

AP_INVALID_TN3270_SUPPORT
The *tn3270_support* parameter for one or more sessions was not set to a valid value.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

DEFINE_TN3270_ASSOCIATION

DEFINE_TN3270_ASSOCIATION 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 verb can be used to define a new association or to overwrite an existing association for a particular display LU.

VCB 構造体

```
typedef struct define_tn3270_association
{
   AP_UINT16        opcode;               /* verb operation code           */
   unsigned char    reserv2;              /* reserved                      */
   unsigned char    format;               /* reserved                      */
   AP_UINT16        primary_rc;           /* primary return code           */
   AP_UINT32        secondary_rc;         /* secondary return code         */
   unsigned char    display_lu_name[8];   /* Display LU name               */
   TN3270_ASSOCIATION_DEF_DATA def_data;  /* association definition        */
} DEFINE_TN3270_ASSOCIATION;

typedef struct tn3270_association_def_data
{
   unsigned char    description[32];      /* description                   */
   unsigned char    reserve0[16];         /* reserved                      */
   unsigned char    printer_lu_name[8];   /* Printer LU name               */
   unsigned char    reserv2[8];           /* reserved                      */
} TN3270_ASSOCIATION_DEF_DATA;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DEFINE_TN3270_ASSOCIATION

display_lu_name
Name of the display LU to be associated with the printer that was specified by the *def_data.printer_lu_name* parameter. This is a type-A EBCDIC string padded on the right with EBCDIC spaces.

The specified display LU should be a display LU defined on the local node, but this is not enforced by the NOF API.

def_data.description
Description of the association being defined. This parameter is optional.

def_data.printer_lu_name
Name of the printer LU to be associated with the display LU that was specified by the *display_lu_name* parameter. This is a type-A EBCDIC string padded on the right with EBCDIC spaces.

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 can 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 verb.

(These rules are not enforced by the NOF API.)

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

ファイル名の変更
指定されたディスプレイ LU 名または指定されたプリンター LU 名のいずれかが、有効な EBCDIC ストリングではありませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

DEFINE_TN3270_DEFAULTS

DEFINE_TN3270_DEFAULTS 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 verb.

VCB 構造体

```
typedef struct define_tn3270_defaults
{
   AP_UINT16        opcode;               /* verb operation code            */
   unsigned char    reserv2;              /* reserved                       */
   unsigned char    format;               /* reserved                       */
   AP_UINT16        primary_rc;           /* primary return code            */
   AP_UINT32        secondary_rc;         /* secondary return code          */
   TN3270_DEFAULTS_DEF_DATA def_data;     /* TN3270 defaults                */
} DEFINE_TN3270_DEFAULTS;
```

```
typedef struct tn3270_defaults_def_data
{
   unsigned char    force_responses;      /* force printer responses?       */
   unsigned char    keepalive_method;     /* method for sending keep-alives */
   AP_UINT32        keepalive_interval;   /* interval between keep-alives   */
   unsigned char    reserv2[32];          /* reserved                       */
} TN3270_DEFAULTS_DEF_DATA;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DEFINE_TN3270_DEFAULTS

def_data.force_responses
Controls client responses on printer sessions. Possible values are:

AP_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 AP_YES to avoid problems.

AP_NO
Request responses matching SNA traffic.

def_data.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:

AP_NONE
Do not send keep-alive messages.

AP_TN3270_NOP
Send Telnet NOP messages.

AP_TN3270_TM
Send Telnet DO TIMING-MARK messages.

def_data.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 when the *keepalive_method* parameter is set to AP_TN3270_NOP or AP_TN3270_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: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_KEEPALIVE
The *keepalive_method* parameter was not set to a valid value.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

DEFINE_TN3270_EXPRESS_LOGON

DEFINE_TN3270_EXPRESS_LOGON sets up the TN3270 Express Logon feature. This feature means that TN3270 client users who connect to CS 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.

VCB 構造体

```
typedef struct define_tn3270_express_logon
{
   AP_UINT16           opcode;             /* verb operation code           */
   unsigned char       reserv2;            /* reserved                      */
   unsigned char       format;             /* reserved                      */
   AP_UINT16           primary_rc;         /* primary return code           */
   AP_UINT32           secondary_rc;       /* secondary return code         */
   unsigned char       dcas_server[256];   /* IP hostname of DCAS server    */
   AP_UINT16           dcas_port;          /* port number to access server  */
   unsigned char       enabled;            /* is Express Logon enabled?     */
   unsigned char       reserv3[33];        /* reserved                      */
} DEFINE_TN3270_EXPRESS_LOGON;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DEFINE_TN3270_EXPRESS_LOGON

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:

AP_YES
The function is enabled, so TN3270 clients can access the host without needing to specify a user ID and password.

AP_NO
The function is not enabled, so TN3270 clients must specify a user ID and password.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

定義済みの LDAP の最大値は 1 つです

DEFINE_TN3270_SSL_LDAP は、 Secure Sockets Layer (SSL) クライアント認証機能で使用するための証明書取り消しリストへのアクセス方法を定義します。 取り消しリストは、外部 LDAP サーバー上で保持され、 TN サーバーまたは TN リダイレクターを使用することが許可されなくなった個々の Telnet クライアントの詳細 (例えば、ユーザーのセキュリティー情報が無許可のパーティーによって検出されたか、またはユーザーが許可された組織に対して機能しなくなったため) を含んでいます。

この機能が使用されている場合、 CS Linux TN サーバーまたは TN リダイレクターに接続する TN3270 クライアントは、証明書 (サーバーを使用する許可を与えられた有効なクライアントとして識別する情報) を提供する必要があります。 その後、サーバーはこの証明書を取り消しリストに照らして検査し、それがまだ有効であることを確認します。

この verb を使用して、 LDAP サーバーへのアクセスの定義、アクセス情報の変更 (ユーザー ID とパスワードの変更など) 、または CS Linux が外部 LDAP サーバー上で失効リストを使用しないことを指定することができます。

この verb は非アクティブ・ノードに対して発行する必要があります。ノードの実行中は、 LDAP サーバーのアクセス情報を変更することはできません。

VCB 構造体

```
typedef struct define_tn3270_ssl_ldap
{
   AP_UINT16           opcode;             /* verb operation code            */
   unsigned char       reserv2;            /* reserved                       */
   unsigned char       format;             /* reserved                       */
   AP_UINT16           primary_rc;         /* primary return code            */
   AP_UINT32           secondary_rc;       /* secondary return code          */
} DEFINE_TN3270_SSL_LDAP;
```

以下のように、 `define_tn3270_ssl_ldap` 構造体は、 VCB の終わりに連結された `tn3270_ssl_ldap_def_data` 構造体のすぐ後に続く必要があります。

```
typedef struct tn3270_ssl_ldap_def_data
{
   AP_UINT16           overlay_size;       /* reserved                       */
   unsigned char       auth_type;          /* type of authorization checking */
   unsigned char       reserv1;            /* reserved                       */
   unsigned char       ldap_addr[256];     /* address of LDAP server         */
   AP_UINT16           ldap_port;          /* port number to access server   */
   unsigned char       ldap_user[1024];    /* user ID on LDAP server         */
   unsigned char       ldap_password[128]; /* password on LDAP server        */
   unsigned char       reserv2[256];       /* reserved                       */
} TN3270_SSL_LDAP_DEF_DATA;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DEFINE_TN3270_SSL_LDAP

def_data.auth_type
Specifies the type of authorization checking performed by the TN Server or TN Redirector. Possible values are:

AP_LOCAL_ONLY
The server checks client certificates locally, but does not use an external certificate revocation list. The parameters *ldap_addr* - *ldap_password* are reserved.

AP_LOCAL_X500
The server checks certificates locally, and also checks against an external certificate revocation list. The remaining parameters in this data structure specify the location of this list.

def_data.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`.

def_data.ldap_port
The TCP/IP port number used to access the LDAP server. The range is 0-65535.

def_data.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.

def_data.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.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc

追加の認証 AUTH_TYPE
auth_type パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

定義済みのリダイレクト

DEFINE_TN_REDIRECT は、 CS Linux の TN リダイレクター機能を使用して、特定の Telnet クライアント (またはすべてのクライアントのデフォルト ・アクセスの詳細) に関するアクセス詳細を定義します。 これを使用して、新規クライアントを定義したり、既存の定義を変更したりすることができます。 (TN3270 Server を使用してクライアントのアクセス詳細を定義するには、 DEFINE_TN3270_ACCESS を使用します)

VCB structure

```
typedef struct define_tn_redirect
{
   AP_UINT16             opcode;           /* verb operation code           */
   unsigned char         reserv2;          /* reserved                      */
   unsigned char         format;           /* reserved                      */
   AP_UINT16             primary_rc;       /* primary return code           */
   AP_UINT32             secondary_rc;     /* secondary return code         */
   TN_REDIRECT_ADDRESS   addr;             /* Uniquely defines record       */
   TN_REDIRECT_DEF_DATA  def_data;         /* verb data                     */
} DEFINE_TN_REDIRECT;
```

```
typedef struct tn_redirect_address
{
   AP_UINT16      default_record;    /* Is this the default record ?         */
   unsigned char  address_format;    /* IP address or fully-qualified name   */
   unsigned char  client_address[256];  /* Client address                    */
   AP_UINT16      port_number;       /* Port number that client connects on  */
   unsigned char  listen_local_address[46];
                                     /* Local addr client connects to        */
   unsigned char  reserved[34];      /* reserved                             */
} TN_REDIRECT_ADDRESS;
```

```
typedef struct tn_redirect_def_data
{
   unsigned char  description[32];       /* Description - null terminated    */
   unsigned char  reserve0[16];          /* Reserved                         */
   unsigned char  cli_ssl_enabled;       /* Is the client session SSL?       */
   unsigned char  host_ssl_enabled;      /* Is the host session SSL?         */
   unsigned char  host_address_format;   /* Type of IP address for the host  */
   unsigned char  reserv1;               /* Reserved                         */
   unsigned char  host_address[256];     /* Host address                     */
   AP_UINT16      host_port_number;      /* Port number to connect to host   */
   unsigned char  cli_conn_security_level;      /* SSL encryption strength   */
   unsigned char  serv_conn_security_level;     /* SSL encryption strength   */
   unsigned char  cli_conn_cert_key_label[80];  /* Key label for certificate */
   unsigned char  serv_conn_cert_key_label[80]; /* Key label for certificate */
   unsigned char  reserved[46];                 /* Reserved                  */
} TN_REDIRECT_DEF_DATA;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DEFINE_TN_REDIRECT

addr.default_record
Specifies whether this verb defines a default record, which will be used by any Telnet client not explicitly identified by a TCP/IP address. If a Telnet client attempts to contact the TN Redirector node, and the user's TCP/IP address does not match any DEFINE_TN_REDIRECT 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:

AP_YES
This verb defines a default record. The *client_address* and *address_format* parameters are reserved.

AP_NO
This verb defines a normal TN Redirector user record.

A default record provides access to the TN Redirector function for any Telnet client 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.

addr.address_format
Specifies the format of the *client_address* parameter. Possible values are:

AP_ADDRESS_IP
IP address (either IPv4 or IPv6)

AP_ADDRESS_FQN
Alias or fully qualified name

addr.client_address
The TCP/IP address of the computer on which the Telnet client runs. This is a null-terminated ASCII string, which 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`).

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 Redirector 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`.

addr.port_number
The number of the server TCP/IP port that the Telnet client 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 `tn_redirect_address` structures 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.

addr.listen_local_address
The address on the local TN Server computer to which TN3270 clients will connect.

- 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, set this parameter to all binary zeros. In this case, any `tn_redirect_address` structure that uses the same *port_number* as this one (for the same *client_address* or a different one) must also have this parameter set to all binary zeros.
- If you need to restrict TN3270 clients to a particular local address, specify it as a null-terminated ASCII string. The address 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`).

  In this case, any `tn_redirect_address` structure 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.

def_data.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 verb, but CS Linux does not make use of it. You can use it to store additional information to help distinguish between users.

def_data.cli_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 NOF verb QUERY_NODE_LIMITS and checking the value of the *ssl_support* parameter.

Possible values are:

AP_NO
The client does not use SSL.

AP_YES
The client uses SSL.

AP_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.

def_data.host_ssl_enabled
Indicates whether the TN Redirector 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 NOF verb QUERY_NODE_LIMITS and checking the value of the *ssl_support* parameter.

Possible values are:

AP_NO
The host does not use SSL.

AP_YES
The host uses SSL.

def_data.host_address_format
Specifies the format of the *host_address* parameter. Possible values are:

AP_ADDRESS_IP
IP address (either IPv4 or IPv6)

AP_ADDRESS_FQN
Alias or fully qualified name

def_data.host_address
The TCP/IP address of the host computer with which the client communicates. This is a null-terminated ASCII string, which can be any of the following; the *host_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 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`.

def_data.host_port_number
The number of the TCP/IP port that the TN Redirector node uses to access the host.

def_data.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_ssl_enabled* parameter is set to `AP_NO`, this parameter is reserved.

Possible values are:

AP_SSL_AUTHENTICATE_MIN
Certificates must be exchanged; encryption is not required (but can be used if the client requests it).

AP_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.

AP_SSL_40_BIT_MIN
Use at least 40-bit encryption.

AP_SSL_56_BIT_MIN
Use at least 56-bit encryption.

AP_SSL_128_BIT_MIN
Use at least 128-bit encryption.

AP_SSL_168_BIT_MIN
Use at least 168-bit encryption.

AP_SSL_256_BIT_MIN
Use at least 256-bit encryption.

Note : Using encryption requires additional software to be installed with CS Linux; see *IBM Communications Server for Data Center Deployment on 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.

def_data.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 CS Linux can support; if the host cannot support the requested level of security or higher, the session will not be started.

If the *host_ssl_enabled* parameter is set to `AP_NO`, this parameter is reserved.

Possible values are:

AP_SSL_AUTHENTICATE_MIN
Certificates must be exchanged; encryption is not required (but can be used if the host requests it).

AP_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.

AP_SSL_40_BIT_MIN
Use at least 40-bit encryption.

AP_SSL_56_BIT_MIN
Use at least 56-bit encryption.

AP_SSL_128_BIT_MIN
Use at least 128-bit encryption.

AP_SSL_168_BIT_MIN
Use at least 168-bit encryption.

AP_SSL_256_BIT_MIN
Use at least 256-bit encryption.

Note : Using encryption requires additional software to be installed with CS Linux; see *IBM Communications Server for Data Center Deployment on 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.

def_data.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 *IBM Communications Server for Data Center Deployment on Linux Quick Beginnings* for more information.

If the *cli_ssl_enabled* parameter is set to AP_NO, this parameter is reserved.

The label is a null-terminated ASCII character string. To use the default SSL certificate and key pair, specified when the SSL keyring database was set up, set this parameter to a null string.

def_data.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 *IBM Communications Server for Data Center Deployment on Linux Quick Beginnings* for more information.

If the *host_ssl_enabled* parameter is set to AP_NO, this parameter is reserved.

The label is a null-terminated ASCII character string. To use the default SSL certificate and key pair, specified when the SSL keyring database was set up, set this parameter to a null string.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_UNKNOWN_CLIENT_ADDRESS
The specified name or alias could not be mapped to a fully qualified name.

AP_CLIENT_CLASH
The combination of port number and fully qualified name (resolved from the *client_address* parameter) clashes with one that has already been defined.

AP_DUPLICATE_PORT_NUMBER
Another TN Redirector 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.

AP_TCPIP_PORT_IN_USE
The TCP/IP port number cannot be used by TN Redirector because it is already in use by a different program.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

定義

DEFINE_TP verb は、 CS Linux がパートナー LU からの着信接続の結果として TP を開始するために必要な情報を提供します。 また、この verb を使用して、以前に定義した TP の 1 つ以上のフィールドを変更することもできます。

起動される TP の標準パラメーターは、呼び出し可能 TP 情報ファイルに定義されています (詳しくは、*IBM Communications Server for Linux* 管理ガイド上のデータ・センター・デプロイメントを参照してください)。 DEFINE_TP は、ファイルに設定できない追加パラメーターを指定する必要がある場合にのみ必要です。会話セキュリティー、確認同期、または会話タイプ (マップまたは基本)に特定のオプションを使用するように TP を制限するか、または任意の時点で実行できる TP のインスタンス数を制限する必要があります。

VCB structure

```
typedef struct define_tp
{
   AP_UINT16        opcode;                /* verb operation code           */
   unsigned char    reserv2;               /* reserved                      */
   unsigned char    format;                /* reserved                      */
   AP_UINT16        primary_rc;            /* primary return code           */
   AP_UINT32        secondary_rc;          /* secondary return code         */
   unsigned char    tp_name[64];           /* TP name                       */
   TP_CHARS         tp_chars;              /* TP characteristics            */
} DEFINE_TP;
```

```
typedef struct tp_chars
{
   unsigned char    description[32];        /* resource description          */
   unsigned char    security_list_name[14]; /* security access list name     */
   unsigned char    reserv1[2];             /* reserved                      */
   unsigned char    conv_type;              /* conversation type             */
   unsigned char    security_rqd;           /* security support              */
   unsigned char    sync_level;             /* synchronisation level support */
   unsigned char    dynamic_load;           /* dynamic load (AP_YES)         */
   unsigned char    enabled;                /* is the TP enabled?            */
   unsigned char    pip_allowed;            /* program initialization        */
                                            /*  parameters supported         */
   unsigned char    reserv3[10];            /* reserved                      */
   AP_UINT16        tp_instance_limit;      /* limit on currently active TP  */
                                            /*  instances                    */
   AP_UINT16        incoming_alloc_timeout; /* incoming allocation timeout   */
   AP_UINT16        rcv_alloc_timeout;      /* receive allocation timeout    */
   AP_UINT16        tp_data_len;            /* reserved                      */
   unsigned char    tp_data[120];           /* reserved                      */
} TP_CHARS;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DEFINE_TP

tp_name
Name of the TP being defined.

tp_chars.description
A null-terminated text string (0-31 characters followed by a null character) describing the TP. This string is for information only; it is stored in the node's configuration file and returned on the QUERY_TP_DEFINITION and QUERY_TP verbs, but CS Linux does not make any other use of it.

tp_chars.security_list_name
Name of the security access list used by this TP (defined using the DEFINE_SECURITY_ACCESS_LIST verb). 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 *tp_chars.security_rqd* parameter must be set to AP_YES.

To specify that the TP is available for use by any user, set this parameter to 14 binary zeros.

tp_chars.conv_type
Specifies the type(s) of conversation supported by this TP. Possible values are:

AP_BASIC
The TP supports only basic conversations.

AP_MAPPED
The TP supports only mapped conversations.

AP_EITHER
The TP supports either basic or mapped conversations.

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

AP_YES
A user ID and password are required to start the TP.

AP_NO
No security information is required.

tp_chars.sync_level
Specifies the values of synchronization level supported by the TP. Possible values are:

AP_NONE
The TP supports only *sync_level* NONE.

AP_CONFIRM_SYNC_LEVEL
The TP supports only *sync_level* CONFIRM.

AP_EITHER
The TP supports either *sync_level* NONE or CONFIRM.

AP_SYNCPT_REQUIRED
The TP supports only *sync_level* SYNCPT (syncpoint is required).

AP_SYNCPT_NEGOTIABLE
The TP supports any of the three *sync_level* values NONE, CONFIRM, and SYNCPT.

tp_chars.dynamic_load
This parameter must be set to AP_YES.

tp_chars.enabled
Specifies whether the TP can be attached successfully. Possible values are:

AP_YES
TP can be attached.

AP_NO
TP cannot be attached.

tp_chars.pip_allowed
Specifies whether the TP can receive Program Initialization Parameters (PIP). Possible values are:

AP_YES
TP can receive PIP.

AP_NO
TP cannot receive PIP.

tp_chars.tp_instance_limit
Limit on the number of instances of this TP that can be active at any one time. A value of zero means no limit.

tp_chars.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.

tp_chars.rcv_alloc_timeout
Number of seconds that a RECEIVE_ALLOCATE verb is queued waiting for an incoming Attach. The value 0 (zero) implies that there is no timeout; the RECEIVE_ALLOCATE verb will be queued indefinitely.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_SYSTEM_TP_CANT_BE_CHANGED
The specified TP name is the name of a TP used internally by CS Linux you cannot define or modify a TP with this name.

AP_INVALID_CONV_TYPE
The *conv_type* parameter was not set to a valid value.

AP_INVALID_SYNC_LEVEL
The *sync_level* parameter was not set to a valid value.

AP_INVALID_DYNAMIC_LOAD
The *dynamic_load* parameter was not set to a valid value.

AP_INVALID_ENABLED
The *enabled* parameter was not set to a valid value.

AP_INVALID_PIP_ALLOWED
The *pip_allowed* parameter was not set to a valid value.

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc
可能な値は次のとおりです

追加のセキュリティー・リストが定義されていない
セキュリティー・リスト名 パラメーターが、定義されたセキュリティー・リスト名と一致しませんでした。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

属性情報の定義

DEFINE_TP_LOAD_INFO は、トランザクション・プログラムがロードされるときに使用される情報を記述する項目を定義または変更します。 An application must issue OPEN_FILE with a requested role of **追加情報** LOAD_INFO before issuing the DEFINE_TP_LOAD_INFO verb.

VCB structure

```
typedef struct define_tp_load_info
{
      AP_UINT16              opcode;              /* verb operation code         */
      unsigned char          reserv2;             /* reserved                    */
      unsigned char          format;              /* reserved                    */
      AP_UINT16              primary_rc;          /* primary return code         */
      AP_UINT32              secondary_rc;        /* secondary return code       */
      unsigned char          tp_name[64];         /* TP name                     */
      unsigned char          lu_alias[8];         /* LU alias                    */
      TP_LOAD_INFO_DEF_DATA  def_data;            /* defined data                */
} DEFINE_TP_LOAD_INFO;
```

```
typedef struct tp_load_info_def_data
{
      unsigned char          description[32];     /* Description                 */
      unsigned char          reserv1[16];         /* reserved                    */
      unsigned char          user_id[64];         /* User ID                     */
      unsigned char          group_id[64];        /* Group ID                    */
      AP_UINT32              timeout;             /* Timeout value               */
      unsigned char          type;                /* TP type                     */
      unsigned char          style;               /* reserved                    */
AP_UINT16              ltv_length;          /* Length of LTV data          */
} TP_LOAD_INFO_DEF_DATA;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DEFINE_TP_LOAD_INFO

tp_name
The TP name of the TP load info entry to be defined. This is a 64-byte EBCDIC string, padded on the right with spaces if the name is shorter than 64 characters.

lu_alias
The LU alias of the TP load info entry to be defined. This is an 8-byte ASCII character string.

Note : This parameter can be used only if the TP is an APPC TP. If the TP is a CPI-C application, this parameter is reserved and must be set to all zeros. 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.

def_data.description
A null-terminated text string (0-32 characters followed by a null character) describing the TP load info. This string is for information only; it is stored in the node's configuration file and returned on the QUERY_TP verb, but CS Linux does not make any other use of it.

def_data.user_id
User ID required to access and run the TP.

def_data.group_id
Group ID required to access and run the TP.

def_data.timeout
Timeout in seconds after the TP is loaded.

def_data.type
Specifies the TP type. Possible values are:

AP_TP_TYPE_QUEUED

AP_TP_TYPE_QUEUED_BROADCAST

AP_TP_TYPE_NON_QUEUED

def_data.ltv_length
Length of the block of LTV data that is appended to this verb. Each LTV structure is specified in TP_LOAD_INFO_LTV.

TP_LOAD_INFO_LTV
The LTV data is specified as a series of non-byte-aligned LTVs each of which consists of the following:

- A 2-byte length field with a maximum value of 258 bytes. This field is in line format and is read or written using NB_PUT_SHORT or NB_GET_SHORT.
- A 1-byte type field set to one of the following possible values:

  AP_TYPE_TP_PATH
  Path. The value string specifies the full path name of the TP executable.

  AP_TYPE_TP_ARGUMENTS
  Arguments. The value string specifies a command-line argument required by the TP.

  AP_TYPE_TP_STDIN
  Standard input. The value string specifies the full path name of the standard input file or device. If this LTV is not specified, the default is /dev/null.

  AP_TYPE_TP_STDOUT
  Standard output. The value string specifies the full path name of the standard output file or device. If this LTV is not specified, the default is /dev/null.

  AP_TYPE_TP_STDERR
  Standard error. The value string specifies the full path name of the standard error file or device. If this LTV is not specified, the default is /dev/null.

  AP_TYPE_TP_ENV
  Environment. The value string specifies an environment variables required by the TP, in the form *VARIABLE* = *VALUE*.

  If the TP is a CPI-C application, note that you cannot set the environment variable APPCLLU using this LTV. The local LU cannot be specified in the TP load information for an automatically-loaded CPI-C application.
- A value field consisting of up to 255 bytes of ASCII data.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

API タイプの付加属性タイプ
タイプ パラメーターが有効な値に設定されていませんでした。

ファイルの長さが AP_INVALID_LTV_LENGTH
LTV 長さ パラメーターが有効な値に設定されていませんでした。

ファイル・タイプの追加タイプ
LTV タイプ パラメーターが有効な値に設定されていませんでした。

付加価値のある LTV_VALUE
LTV 値 パラメーターに、無効なデータが含まれていました。

AP_INVALID_TP_STYLE
TP 文体 パラメーターに、無効な値が含まれています。

ファイル名の変更
TP 名称 パラメーターには EBCDIC スペースが含まれています

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

ユーザー ID のパスワード

DEFINE_USERID_PASSWORD は、 APPC および CPI-C 会話セキュリティーで使用するためのユーザー ID とパスワードのペアを定義するか、または定義済みのユーザー ID とパスワードのプロファイルを追加します。

VCB structure

```
typedef struct define_userid_password
{
    AP_UINT16              opcode;             /* verb operation code        */
    unsigned char          reserv2;            /* reserved                   */
    unsigned char          format;             /* reserved                   */
    AP_UINT16              primary_rc;         /* primary return code        */
    AP_UINT32              secondary_rc;       /* secondary return code      */
    AP_UINT16              define_type;        /* what the define type is    */
    unsigned char          user_id[10];        /* user id                    */
    unsigned char          reserv3[8];         /* reserved                   */
    USERID_PASSWORD_CHARS  password_chars;     /* password characteristics   */
} DEFINE_USERID_PASSWORD;
```

```
typedef struct userid_password_chars
{
    unsigned char          description[32];    /* resource description       */
    unsigned char          reserv2[16];        /* reserved                   */
    AP_UINT16              profile_count;      /* number of profiles         */
    AP_UINT16              reserv1;            /* reserved                   */
    unsigned char          password[10];       /* password                   */
    unsigned char          profiles[10][10];   /* profiles                   */
} USERID_PASSWORD_CHARS;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
ユーザー ID ユーザー ID のパスワード

define_type
この verb をどのように使用するかを指定します。 可能な値は次のとおりです

追加アドレス・ユーザー
新規ユーザーを追加するか、既存のユーザーのパスワードを変更します。

エイプ・アドレス・プロファイル
既存のユーザーのプロファイルに追加します。

ユーザー *ID*
ユーザー ID。 これは 10 バイトのタイプ AE の EBCDIC 文字ストリングで、右側に EBCDIC のスペースが埋め込まれます。

一部の CPI-C インプリメンテーションでは、ユーザー ID の長さが最大 8 文字になります。 9 文字または 10 文字のユーザー ID を指定すると、他のシステムで実行されている CPI-C アプリケーションは、このユーザー ID とパスワードを使用して CS Linux システム上のアプリケーションにアクセスできなくなる可能性があります。

パスワードの説明 . 説明
ユーザー ID とパスワードを記述したヌルで終了するテキスト・ストリング (0 から 31 文字の後にヌル文字を続けたストリング) 。 このストリングは情報専用です。このストリングはノードの構成ファイルに保管され、 QUERY_USERID_PASSWORD verb で戻されますが、 CS Linux ではそれ以外の使用は行われません。

パスワード・チャート . *profile_count*
プロファイルの数。 通常、このパラメーターはゼロに設定されます。詳しくは、以下の パスワード・プロファイル . プロファイル を参照してください

パスワード・文字 . パスワード
ユーザーのパスワード。 これは 10 バイトのタイプ AE の EBCDIC 文字ストリングで、右側に EBCDIC のスペースが埋め込まれます。

一部の CPI-C インプリメンテーションでは、パスワードの長さが最大 8 文字になります。 9 文字または 10 文字のパスワードを指定すると、他のシステムで実行されている CPI-C アプリケーションは、このユーザー ID とパスワードを使用して CS Linux システム上のアプリケーションにアクセスできなくなる場合があります。

このパラメーターに対してアプリケーションが提供する値が、暗号化されたバージョンのパスワードですぐに置換されます。 そのため、 パスワード・文字 . パスワード パラメーターに指定された値は、決して書き出されません。

パスワード・プロファイル . プロファイル
ユーザー ID とパスワードに関連付けられたプロファイル名。 これらはそれぞれ 10 バイトのタイプ AE の EBCDIC 文字ストリングで、右側に EBCDIC のスペースが埋め込まれます。

リモート TP がこのユーザー ID とパスワードを使用してローカル TP に接続し、 Attach 上でプロファイルを指定する場合、これはここで定義されているプロファイル名の 1 つと一致する必要があります。 リモート・システム管理者に確認して、プロファイルが使用されるかどうかを判別します。使用されるプロファイルごとに、この verb で プロファイル パラメーターの 1 つとしてプロファイル名を指定します。 ほとんどの場合、プロファイル名は使用されないため、この verb ではプロファイル名を指定する必要はありません。 パスワード・チャート . *profile_count* をゼロに設定し、プロファイルを指定しないでください。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_PASSWORD
The *password* parameter contained a character that was not valid.

AP_INVALID_PROFILE
One or more of the specified profiles was not valid.

AP_INVALID_UPDATE_TYPE
The *define_type* parameter was not set to a valid value.

AP_INVALID_USERID
The *user_id* parameter contained a character that was not valid.

AP_NO_PROFILES
The verb was used to add profiles to an existing user, but no profiles were specified.

AP_TOO_MANY_PROFILES
The *profile_count* parameter was not set to a valid value.

AP_UNKNOWN_USER
The verb was used to add profiles to an existing user, but the *user_id* parameter did not match an existing user ID.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

区切り文字の削除のノード・ノード

DELETE_ADJACENT_LEN_NODE は、ノード・ディレクトリー・データベース内のエントリーを隣接 LEN ノードおよびその関連 LU に対して除去するか、 LEN ノード自体を除去せずに LEN ノードの LU エントリーを除去します。 これは、 LEN ノードおよびその関連 LU に一連の DELETE_DIRECTORY_ENTRY verb を発行するのと同等です。

VCB structure

```
typedef struct delete_adjacent_len_node
{
     AP_UINT16          opcode;              /* verb operation code          */
     unsigned char      reserv2;             /* reserved                     */
     unsigned char      format;              /* reserved                     */
     AP_UINT16          primary_rc;          /* primary return code          */
     AP_UINT32          secondary_rc;        /* secondary return code        */
     unsigned char      cp_name[17];         /* CP name                      */
     unsigned char      num_of_lus;          /* number of LUs                */
     unsigned char      lu_names[10][8];     /* LU names                     */
     unsigned char      wildcard_lus;        /* wildcard LUs                 */
} DELETE_ADJACENT_LEN_NODE;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
AP_DELETE_ADJACENT_LEN_NODE

cp_name
隣接 LEN ノード内の CP の完全修飾名。 この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、

EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

lus* の *num_of_lus
削除する LU の数 (1 から 10 までの範囲) 。 LEN ノード定義全体を削除するには、ゼロを指定します。

***LU* 名**
LEN ノード上の削除される LU の名前。 各名前は 8 バイトのタイプ A の EBCDIC 文字ストリングで、右側には EBCDIC のスペースが埋め込まれます。 LEN ノード定義全体を削除する場合は、 LU 名を指定しないでください (*lus* の *num_of_lus* がゼロの場合) 。

名前の最初の文字だけを指定することによって、複数の LU 名に一致するように "ワイルドカード" LU 名を指定することができます。 例えば、ワイルドカード LU 名 APPN.LU は APPN.LUNAME または APPN.LU01 と一致します (ただし、 APPN.NAMELU とは一致しません)。 ただし、単一の verb で指定されるすべての LU 名は、以下の ワイルドカード *(d_lus)* パラメーターで定義されているとおりに、同じタイプ (ワイルドカードまたは明示) でなければなりません。 同じ LEN ノードから両方のタイプの LU 名を除去するには、複数の DELETE_ADJACENT_LEN_NODE verb を使用します。

ワイルドカード *(d_lus)*
指定された LU 名がワイルドカード・エントリーであるか明示的な LU 名であるかを示します。 可能な値は次のとおりです

類人猿
指定された LU 名はワイルドカード・エントリーです。

アプ・ノー
指定された LU 名は明示的な項目です。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

ファイルの検証 CP_NAME
cp_name パラメーターに、無効な文字が含まれていました。

ファイル名の変更
指定された LU 名の 1 つ以上に、無効な文字が含まれていました。

類型の無効です。
lus の *num_of_lus* パラメーターが有効範囲内にありませんでした。

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc
可能な値は次のとおりです

ファイルの検証 CP_NAME
指定された CP 名は存在しません。

ファイル名の変更
指定された LU 名の 1 つ以上が存在しません。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

DELETE_BACKUP

An application uses this verb to delete a server from the list of backup servers in the `sna.net` file, so that this server can no longer act as the controlling configuration file server.

You can use this verb to delete any server in the list, including the controller 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 controller 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 verb must be issued to the `sna.net` file.

VCB structure

```
typedef struct delete_backup
{
     AP_UINT16          opcode;              /* verb operation code         */
     unsigned char      reserv2;             /* reserved                    */
     unsigned char      format;              /* reserved                    */
     AP_UINT16          primary_rc;          /* primary return code         */
     AP_UINT32          secondary_rc;        /* secondary return code       */
     unsigned char      backup_name[128];    /* name of server to delete    */
     unsigned char      reserv3[4];          /* reserved                    */
} DELETE_BACKUP;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
AP_DELETE_BACKUP

バックアップ名
バックアップ・サーバーのリストから削除されるサーバーの名前。

サーバー名に . (ピリオド) 文字が含まれている場合、 CS Linux は、サーバー名が完全修飾名であると見なします。それ以外の場合は、 DNS ルックアップを実行してサーバー名を判別します。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

secondary_rc
未使用。

戻りパラメーター: 状態チェック

状態チェックのために verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc
可能な値は次のとおりです

追加されたレコードが見つかりません
指定されたサーバー名がファイルにリストされていません。

AP_CANT_DELETE_LAST_BACKUP
リストからサーバー名を削除することはできません。これは、 SNA ソフトウェアが実行されている唯一のサーバーであるためです (したがって、現在コントローラーとして機能している唯一のサーバーのみ)。 それを削除する前に、リストされている他の 1 つ以上のサーバーで SNA ソフトウェアを開始するか、または 1 つ以上の新規バックアップ・サーバー (ADD_BACKUP を使用) を追加し、これらのサーバーで SNA ソフトウェアが開始されていることを確認してください。

ファイルの検証ターゲット
NOF API 呼び出しのターゲット・ハンドルに、構成ファイルまたはノードが指定されています。 This verb must be issued to the **スネネット** ファイル。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **状態検査の追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

削除 (CN)

DELETE_CN は、接続ネットワークを削除するか、または選択されたポートを接続ネットワークから削除します。

この verb は、 LEN ノードではなく、ネットワーク・ノードまたはエンド・ノードでのみ有効です。

VCB 構造体

```
typedef struct delete_cn
{
     AP_UINT16          opcode;             /* verb operation code            */
     unsigned char      reserv2;            /* reserved                       */
     unsigned char      format;             /* reserved                       */
     AP_UINT16          primary_rc;         /* primary return code            */
     AP_UINT32          secondary_rc;       /* secondary return code          */
     unsigned char      fqcn_name[17];      /* name of Connection Network     */
     unsigned char      reserv1;            /* reserved                       */
     AP_UINT16          num_ports;          /* number of ports to delete      */
     unsigned char      port_name[8][8];    /* names of ports to delete       */
} DELETE_CN;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
追加の削除 （ CN ）

fqcn_name
接続ネットワークの完全修飾名。 この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

ポート数
接続ネットワークを削除する場合はゼロを指定し、接続ネットワークを削除せずにポートを除去する場合には、削除するポートの数を指定します。

ポート名
ポートを除去する場合 (ポート数 がゼロ以外の場合)、削除するポートの名前を指定します。 各ポート名は 8 バイトの ASCII ストリングで、名前が 8 バイトより短い場合は、右側にスペースが埋め込ま

れます。 接続ネットワークを削除する場合 (ポート数 がゼロの場合) 、これらの名前は 2 進ゼロに設定する必要があります。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

ファイル・ ファイル名の変更
指定された完全修飾 CN 名が、定義された CN 名と一致しませんでした。

指定された AP_INVALID_NUM_PORTS_ネス
ポート数 パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: function not supported

If the verb does not execute successfully because the local node is a LEN node, CS Linux returns the following parameters:

primary_rc

AP_FUNCTION_NOT_SUPPORTED
The local node is a LEN node. This verb is valid only at a network node or an end node.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

DELETE_COS

DELETE_COS deletes a class of service entry. Only locally defined classes of service can be deleted; the default classes of service 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 DEFINE_NODE) and the configuration includes modes that are mapped to the COS that you are deleting, CS Linux will remap these modes to the default COS (specified by a DEFINE_MODE verb with a null mode name) or to the SNA-defined COS #CONNECT if no default COS is specified.

VCB 構造体

```
typedef struct delete_cos
{
     AP_UINT16          opcode;             /* verb operation code          */
     unsigned char      reserv2;            /* reserved                     */
     unsigned char      format;             /* reserved                     */
     AP_UINT16          primary_rc;         /* primary return code          */
     AP_UINT32          secondary_rc;       /* secondary return code        */
     unsigned char      cos_name[8];        /* class of service name        */
} DELETE_COS;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DELETE_COS

cos_name
Class of service name. This is an 8-byte type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

余弦の名前を定義しない （ _L ）
指定された名前は、 CS Linux システムで定義された COS の名前ではありません。

削除された AP_SNA_DEFD_COST_BE_抹消されたもの
指定された名前は、 SNA が定義したサービス・クラスの 1 つの名前であり、削除できません。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

DELETE_CPIC_SIDE_INFO

This verb deletes an entry from the side information table.

Note the difference between this verb and the CPI-C function `Delete_CPIC_Side_Information`. This verb modifies a configuration file, so that it affects all CS Linux CPI-C applications. The CPI-C function modifies the application's own copy in memory of the side information table, and does not affect any other CPI-C applications.

This verb must be issued to the domain configuration file.

VCB structure

```
typedef struct delete_cpic_side_info
{
     AP_UINT16          opcode;              /* verb operation code          */
     unsigned char      reserv2;             /* reserved                     */
     unsigned char      format;              /* reserved                     */
     AP_UINT16          primary_rc;          /* primary return code          */
     AP_UINT32          secondary_rc;        /* secondary return code        */
     unsigned char      reserv2a[8];         /* reserved                     */
     unsigned char      sym_dest_name[8];    /* Symbolic destination name    */
} DELETE_CPIC_SIDE_INFO;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DELETE_CPIC_SIDE_INFO

sym_dest_name
Symbolic destination name which identifies the side information entry. This is an 8-byte ASCII string, consisting of uppercase A-Z and digits 0-9, padded on the right with spaces if necessary.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc

ファイル名の宛先名が表示される
sym_dest_name パラメーターが、定義済みの CPI-C サイド情報エントリーの名前ではありませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **状態検査の追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

DELETE_DIRECTORY_ENTRY

DELETE_DIRECTORY_ENTRY 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.

If the entry for a parent resource is deleted, then all entries for child resources associated with it are also deleted. For example, if 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.

VCB 構造体

```
typedef struct delete_directory_entry
{
   AP_UINT16          opcode;              /* verb operation code            */
   unsigned char      reserv2;             /* reserved                       */
   unsigned char      format;              /* reserved                       */
   AP_UINT16          primary_rc;          /* primary return code            */
   AP_UINT32          secondary_rc;        /* secondary return code          */
   unsigned char      resource_name[17];   /* fully qualified resource name  */
   unsigned char      reserv3;             /* reserved                       */
   AP_UINT16          resource_type;       /* resource type                  */
} DELETE_DIRECTORY_ENTRY;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DELETE_DIRECTORY_ENTRY

resource_name
Fully qualified name of the resource to be deleted. The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

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

AP_ENCP_RESOURCE
End node or LEN node

AP_NNCP_RESOURCE
Network node

AP_LU_RESOURCE
LU

AP_WILDCARD_LU_RESOURCE
Wildcard LU name.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_FQ_LU_NAME
The *resource_name* parameter was not the name of a defined directory entry.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc

AP_CANT_DELETE_ADJ_ENDNODE
指定された項目はエンド・ノード用であり、この verb が発行された先のノードがそのネットワーク・ノード・サーバーです。 このエンド・ノード項目は削除できません。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **状態検査の追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

削除時の削除

DELETE_DLC は、 DLC を削除します。 この verb は、以下を削除します。

- DLC に関連したすべてのポート、リンク・ステーション、および接続ネットワーク TG
- DLC 上の LS に関連付けられたすべての PU 、これらの PU が所有するすべての LU 、およびこれらの LU に関連するすべての LU-LU パスワード。

VCB structure

```
typedef struct delete_dlc
{
     AP_UINT16           opcode;             /* verb operation code         */
     unsigned char       reserv2;            /* reserved                    */
     unsigned char       format;             /* reserved                    */
     AP_UINT16           primary_rc;         /* primary return code         */
     AP_UINT32           secondary_rc;       /* secondary return code       */
     unsigned char       dlc_name[8];        /* name of DLC                 */
} DELETE_DLC;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DELETE_DLC

dlc_name
Name of DLC to be deleted. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc

ファイル名の変更ができない
指定された DLC 名は、 CS Linux システム上で定義された DLC の名前ではありませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc

追加 DLC_アクティブ
DLC は、現在活動状態であるため、削除できません。 STOP_DLC verb を使用して、削除を試みる前に停止してください。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **状態検査の追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

DELETE_DOWNSTREAM_LU

This verb is used to delete a downstream LU.

VCB 構造体

```
typedef struct delete_downstream_lu
{
     AP_UINT16          opcode;              /* verb operation code        */
     unsigned char      reserv2;             /* reserved                   */
     unsigned char      format;              /* reserved                   */
     AP_UINT16          primary_rc;          /* primary return code        */
     AP_UINT32          secondary_rc;        /* secondary return code      */
     unsigned char      dslu_name[8];        /* Downstream LU name         */
} DELETE_DOWNSTREAM_LU;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DELETE_DOWNSTREAM_LU

dslu_name
Name of the downstream LU that is being deleted. This is an 8-byte type A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces if the name is shorter than 8 characters.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc

ファイル名の変更
dslu_name パラメーターに、無効な文字が含まれていました。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: state check

If the verb does not execute because of a state error, CS Linux returns the following parameters.

primary_rc
AP_STATE_CHECK

secondary_rc
Possible values are:

AP_INVALID_LU_NAME
The *dslu_name* parameter did not match any defined downstream LU name.

AP_DSLU_ACTIVE
The LU cannot be deleted because it is currently active.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_STATE_CHECK, which are common to all NOF verbs.

Returned parameters: function not supported

If the verb does not execute because the node's configuration does not support it, CS Linux returns the following parameters:

primary_rc

AP_FUNCTION_NOT_SUPPORTED
The local node does not support SNA gateway; this is defined by the *pu_conc_support* parameter on the DEFINE_NODE verb.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

削除された削除の範囲

この verb は、ダウンストリーム LU の範囲を削除するために使用します。

この verb に提供されるパラメーターには、 LU のベース名と NAU アドレスの範囲が含まれます。 削除される LU 名は、基本名と NAU アドレスを結合することによって判別されます。 例えば、 LUNME のベース名と NAU 範囲の 11 から 14 を組み合わせると、 LU LUNME011、 LUNME012、 LUNME013、および LUNME014 が削除されます。

指定された範囲内の名前を持つすべての LU が削除されます。範囲内の 1 つ以上の名前が存在しない場合、 CS Linux はエラーを戻しません。

VCB structure

```
typedef struct delete_downstream_lu_range
{
    AP_UINT16         opcode;              /* verb operation code           */
    unsigned char     reserv2;             /* reserved                      */
    unsigned char     format;              /* reserved                      */
    AP_UINT16         primary_rc;          /* primary return code           */
    AP_UINT32         secondary_rc;        /* secondary return code         */
    unsigned char     dslu_base_name[5];   /* LU base name                  */
    unsigned char     min_nau;             /* Minimum NAU address in range  */
    unsigned char     max_nau;             /* Maximum NAU address in range  */
} DELETE_DOWNSTREAM_LU_RANGE;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
追加された削除の削除範囲

dslu_base_name
LU の名前のベース名。 これは 5 バイトのタイプ A の EBCDIC ストリング (文字で始まる) で、ベース名が 5 文字より短い場合は、右側に EBCDIC のスペースが埋め込まれます。 CS Linux は、各 NAU アドレスの 3 桁の 10 進値をこの名前に付加することによって、削除される LU の名前を判別します。

ミンナウ
1-255 の範囲内の最初の LU の NAU アドレス。

最大値
1-255 の範囲内の最後の LU の NAU アドレス。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

追加の無効メールアドレス
ミンナウ パラメーターまたは 最大値 パラメーターが無効でした。

ファイル名の変更
dslu_base_name パラメーターに、無効な文字が含まれていました。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc
可能な値は次のとおりです

ファイル名の変更
指定された範囲内に名前を持つ LU が定義されていません。

アクティブの追加 (_SLU_ACTIVE)
この範囲内の 1 つ以上の LU は、現在アクティブであるため、削除できません。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **状態検査の追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: function not supported

If the verb does not execute because the node's configuration does not support it, CS Linux returns the following parameters:

primary_rc

AP_FUNCTION_NOT_SUPPORTED
The local node does not support SNA gateway; this is defined by the *pu_conc_support* parameter on the DEFINE_NODE verb.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

DSPU_TEMPLATE の削除

DELETE_DSPU_TEMPLATE verb は、前に DEFINE_DSPU_TEMPLATE verb を使用して定義された特定のダウンストリーム物理装置 (DSPU) テンプレートを削除するか、または DSPU テンプレートから 1 つ以上のダウンストリーム LU (DSLU) テンプレートを削除します。

VCB 構造体

```
typedef struct delete_dspu_template
{
    AP_UINT16         opcode;                /* verb operation code          */
    unsigned char     reserv2;               /* reserved                     */
    unsigned char     format;                /* reserved                     */
    AP_UINT16         primary_rc;            /* primary return code          */
    AP_UINT32         secondary_rc;          /* secondary return code        */
    unsigned char     template_name[8];      /* name of template             */
    AP_UINT16         num_of_dslu_templates; /* number of dslu templates     */
    unsigned char     reserv1[10];           /* reserved                     */
} DELETE_DSPU_TEMPLATE;
```

```
typedef struct dslu_template
}
    unsigned char     min_nau;               /* Minimum NAU address in range */
    unsigned char     max_nau;               /* Maximum NAU address in range */
    unsigned char     allow_timeout;         /* Allow timeout of host LU?    */
    unsigned char     delayed_logon;         /* Allow delayed logon to host  */
                                             /*  LU                          */
    unsigned char     reserv1[8];            /* reserved                     */
    unsigned char     host_lu[8];            /* Host LU or Pool name         */
} DSLU_TEMPLATE;
```

提供されるパラメーター

提供されるパラメーター :

オペコード
追加テンプレート· テンプレート

テンプレート名
削除する DSPU テンプレートの名前、または削除する DSLU テンプレートが入っている DSPU テンプレートの名前。 1 から 8 桁のローカル表示可能文字を指定します

***d_of_dslu_*テンプレートの数**
削除される DSLU テンプレートの数。 1-255 の範囲の値を指定するか、または 0 (ゼロ) を指定して DSPU テンプレート全体を削除してください。

削除される DSLU テンプレートごとに、 *d_of_dslu_*テンプレートの数で指定された数まで、以下のパラメーターを含む DELETE_DSPU_TEMPLATE 構造体の終わりに DSLU_TEMPLATE 構造体を付加します。

ミンナウ
削除される DSLU テンプレートの範囲内の最小 NAU アドレス。 1-255 の範囲の値を指定してください。

最大値
削除される DSLU テンプレートの範囲内の最大 NAU アドレス。 1-255 の範囲の値を指定してください。

allow_timeout
セッションがホスト LU 定義で指定されたタイムアウト期間に非アクティブのままである場合に、このダウンストリーム LU が使用するホスト LU を CS Linux がタイムアウトにすることを許可するかどうかを指定します。 可能な値は次のとおりです

類人猿
CS Linux は、このダウンストリーム LU が使用するホスト LU のタイムアウトを許可されます。

アプ・ノー
CS Linux は、このダウンストリーム LU が使用するホスト LU をタイムアウトにすることはできません

ログオンの *delayed_logon*
ダウンストリーム LU から最初のデータが受信されるまで、ダウンストリーム LU をホスト LU に接続する CS Linux の遅延を指定します。 代わりに、シミュレートされたログオン画面がダウンストリーム LU に送信されます。 可能な値は次のとおりです

類人猿
ダウンストリーム LU から最初のデータが受信されるまで、ダウンストリーム LU をホスト LU に接続する CS Linux の遅延。

アプ・ノー
CS Linux は、ダウンストリーム LU から最初のデータが受信されるまで、ダウンストリーム LU とホスト LU の接続を遅延しません。

ホスト *(lu)*
範囲内のすべてのダウンストリーム LU がマップされるホスト LU またはホスト LU プールの名前。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

アプリケーション・テンプレート名の変更
テンプレート名 パラメーターで指定されたテンプレートが有効ではありませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

フォーカス・ポイントの削除

DELETE_FOCAL_POINT verb は、指定された MS カテゴリーのフォーカル・ポイントの定義 (そのカテゴリーまたはバックアップ・フォーカル・ポイントのメイン・フォーカル・ポイントのいずれか) を除去します。 定義済みのフォーカル・ポイント・アプリケーションがアクティブで、そのカテゴリーの現在のフォーカル・ポイントとして機能している場合、 CS Linux は、そのフォーカル・ポイントに MS_CAPABILITIES メッセージを送信して、フォーカル・ポイントとして機能しなくなるように、そのフォーカル・ポイントに取り消しをします。

VCB 構造体

```
typedef struct delete_focal_point
{
     AP_UINT16         opcode;           /* verb operation code            */
     unsigned char     reserv2;          /* reserved                       */
     unsigned char     format;           /* reserved                       */
     AP_UINT16         primary_rc;       /* primary return code            */
     AP_UINT32         secondary_rc;     /* secondary return code          */
     unsigned char     reserved;         /* reserved                       */
     unsigned char     ms_category[8];   /* management services category   */
     unsigned char     type;             /* type of focal point            */
} DELETE_FOCAL_POINT;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DELETE_FOCAL_POINT

ms_category
Management Services category. This may be either one of the category names specified in the MS Discipline-Specific Application Programs table of *Systems Network Architecture: Management Services Reference* (see the Bibliography), padded with EBCDIC spaces (0x40), or a user-defined category. A user-defined category name is an 8-byte type-1134 EBCDIC string, padded with EBCDIC spaces (0x40) if necessary.

type
Specifies the type of the focal point that is being deleted. Possible values are:

AP_ACTIVE
The currently active focal point (which may be of any type) is revoked.

AP_IMPLICIT
The implicit definition (defined using DEFINE_FOCAL_POINT with backup set to AP_NO) is removed. If this focal point is currently active, then it is revoked.

AP_BACKUP
The backup definition (defined using DEFINE_FOCAL_POINT with backup set to AP_YES) is removed. If this focal point is currently active, then it is revoked.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

ファイル・ カテゴリー名の変更
指定されたカテゴリー名に、無効な文字が含まれていました。

追加の無効タイプ
タイプ パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: 関数はサポートされません

ローカル・ノード構成がそれをサポートしていないために verb が正常に実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc

付加機能がサポートされていません
ローカル・ノードは MS ネットワーク管理機能をサポートしていません。これは、 DEFINE_NODE verb の *mds_supported* パラメーターによって定義されます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

内部の削除 (_C)

DELETE_INTERNAL_PU は、 DLUR 提供のローカル PU を削除します。 PU は、アクティブな SSCP-PU セッションを持たない場合にのみ削除できます。

VCB 構造体

```
typedef struct delete_internal_pu
{
     AP_UINT16          opcode;          /* verb operation code            */
     unsigned char      reserv2;         /* reserved                       */
     unsigned char      format;          /* reserved                       */
     AP_UINT16          primary_rc;      /* primary return code            */
     AP_UINT32          secondary_rc;    /* secondary return code          */
     unsigned char      pu_name[8];      /* internal PU name               */
} DELETE_INTERNAL_PU;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DELETE_INTERNAL_PU

pu_name
Name of the internal PU that is being deleted. This is an 8-byte type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc

AP_INVALID_PU_NAME
The *pu_name* parameter was not the name of a defined internal PU.

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc
可能な値は次のとおりです

AP_PU_NOT_RESET
PU は活動状態の PU-SSCP セッションがまだあるため、削除できません。

エイプ無効 PU_TYPE
指定された PU はリモート PU であり、内部 PU ではありません。

戻りパラメーター: 関数はサポートされません

ノードの構成がそれをサポートしていないために verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc

付加機能がサポートされていません
ノードは DLUR をサポートしていません。これは、 DEFINE_NODE の *dlur_support* パラメーターによって定義されます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

DELETE_LOCAL_LU

The DELETE_LOCAL_LU verb deletes a local LU, and also deletes any LU-LU passwords associated with the local LU.

VCB 構造体

```
typedef struct delete_local_lu
{
     AP_UINT16           opcode;             /* verb operation code         */
     unsigned char       reserv2;            /* reserved                    */
     unsigned char       format;             /* reserved                    */
     AP_UINT16           primary_rc;         /* primary return code         */
     AP_UINT32           secondary_rc;       /* secondary return code       */
     unsigned char       lu_name[8];         /* local LU name               */
} DELETE_LOCAL_LU;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
アプデレット LOCAL_LU

lu_name
削除されるローカル LU の名前。 これは 8 バイトのタイプ A の EBCDIC ストリング (文字で始まる) で、名前が 8 文字より短い場合は、右側に EBCDIC のスペースが埋め込まれます。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_CANT_DELETE_CP_LU
The supplied LU name was blank (indicating the LU associated with the CP); this LU cannot be deleted.

AP_INVALID_LU_NAME
The supplied LU name is not the name of a local LU defined on the CS Linux system.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

DELETE_LS

DELETE_LS deletes a defined Link Station (LS). This verb 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.

VCB 構造体

```
typedef struct delete_ls
{
     AP_UINT16          opcode;            /* verb operation code            */
     unsigned char      reserv2;           /* reserved                       */
     unsigned char      format;            /* reserved                       */
     AP_UINT16          primary_rc;        /* primary return code            */
     AP_UINT32          secondary_rc;      /* secondary return code          */
     unsigned char      ls_name[8];        /* name of link station           */
} DELETE_LS;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
追加の削除の数

ls_name
削除されるリンク・ステーションの名前。 これは 8 バイトの ASCII ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc

AP_INVALID_LINK_NAME
The supplied LS name contains a character that was not valid.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: state check

If the verb does not execute because of a state error, CS Linux returns the following parameters.

primary_rc
AP_STATE_CHECK

secondary_rc
Possible values are:

AP_LS_ACTIVE
The LS cannot be deleted because it is currently active.

AP_INVALID_LINK_NAME
The supplied LS name is not the name of an LS defined on the CS Linux system.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_STATE_CHECK, which are common to all NOF verbs.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

DELETE_LS_ROUTING

The DELETE_LS_ROUTING verb deletes the association of a partner LU to a link station that was previously defined using the DEFINE_LS_ROUTING verb.

VCB 構造体

```
typedef struct delete_ls_routing
{
    AP_UINT16         opcode;               /* verb operation code          */
    unsigned char     reserv2;              /* reserved                     */
    unsigned char     format;               /* reserved                     */
    AP_UINT16         primary_rc;           /* primary return code          */
    AP_UINT32         secondary_rc;         /* secondary return code        */
    unsigned char     lu_name[8];           /* LU Name                      */
    unsigned char     lu_alias[8];          /* reserved                     */
    unsigned char     fq_partner_lu[17];    /* partner lu name              */
    unsigned char     wildcard_fqplu;       /* wildcard partner LU flag     */
    unsigned char     reserv3[2];           /* reserved                     */
} DELETE_LS_ROUTING;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
AP_DELETE_LS_ルーティング

lu_name
パートナー LU と通信したローカル LU の名前 (*fq_partner_lu* パラメーターで指定される) 。 1 から 8 桁のローカル表示可能文字を指定します

fq_partner_lu
ローカル LU の LS 経路指定データから除去されるパートナー LU の完全修飾名。 1 から 8 文字のネットワーク名、ピリオド、 1 から 8 文字のパートナー LU 名で構成される、 3 から 17 桁のローカル表示可能文字を指定します。

ワイルドカード・エントリーを削除するには、エントリーの定義に使用したものと同じワイルドカード LU 名を指定します。 ワイルドカードを使用して、明示的に定義された複数の項目を削除することはできません。

ワイルドカード・ディスク
ワイルドカード・パートナー LU フラグ。 *fq_partner_lu* パラメーターに完全ワイルドカードまたは部分ワイルドカードが含まれているかどうかを示します。 このフラグは、ワイルドカード項目を削除するために使用されます。ワイルドカードを使用して、明示的に定義された複数の項目を削除することはできません 可能な値は次のとおりです

類人猿
fq_partner_lu パラメーターにワイルドカード・エントリーが含まれています。

アプ・ノー
fq_partner_lu パラメーターにワイルドカード・エントリーが含まれていません。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_LOCAL_LU
The *lu_name* parameter contained a character that was not valid.

AP_INVALID_PARTNER_LU
The *fq_partner_lu* parameter contained a character that was not valid.

AP_INVALID_WILDCARD_NAME
The *wildcard_fqplu* parameter was set to AP_YES, but the *fq_partner_lu* parameter was not a valid wildcard name.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc
可能な値は次のとおりです

追加のローカル・ローカル・ルー
lu_name パラメーターが、既存の LS 経路指定レコードと一致しませんでした。

初期化さができない
fq_partner_lu パラメーターが、指定されたローカル LU の既存の LS 経路指定レコードと一致しませんでした。

ファイル・ワイルドカード名を使用できます
ワイルドカード・ディスク パラメーターが **そうだ**に設定されましたが、一致するエントリーが見つかりませんでした。

ソース・リソース名が無効です
指定されたパラメーターに一致する LS 経路指定項目が見つかりませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **状態検査の追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

DELETE_LU62_TIMEOUT

The DELETE_LU62_TIMEOUT verb deletes a definition of an LU type 6.2 session timeout that was defined previously with a DEFINE_LU62_TIMEOUT verb.

VCB 構造体

```
typedef struct delete_lu62_timeout
{
    AP_UINT16         opcode;                /* verb operation code        */
    unsigned char     reserv2;               /* reserved                   */
    unsigned char     format;                /* reserved                   */
    AP_UINT16         primary_rc;            /* primary return code        */
    AP_UINT32         secondary_rc;          /* secondary return code      */
    unsigned char     resource_type;         /* resource type              */
    unsigned char     resource_name[17];     /* resource name              */
} DELETE_LU62_TIMEOUT;
```

Supplied parameters

Supplied parameters are:

opcode
AP_DELETE_LU62_TIMEOUT

resource_type
Specifies the type of timeout being deleted. Possible values are:

AP_GLOBAL_TIMEOUT
Delete timeouts that apply to all LU 6.2 sessions for the local node.

AP_LOCAL_LU_TIMEOUT
Delete timeouts that apply to all LU 6.2 sessions for the local LU specified in the *resource_name* parameter.

AP_PARTNER_LU_TIMEOUT
Delete timeouts that apply to all LU 6.2 sessions to the partner LU specified in the *resource_name* parameter.

AP_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 AP_GLOBAL_TIMEOUT, do not specify this parameter.

- If *resource_type* is set to AP_LOCAL_LU_TIMEOUT, specify 1-8 locally displayable type-A characters as a local LU name.
- If *resource_type* is set to AP_PARTNER_LU_TIMEOUT, specify the fully qualified name of the partner LU as follows: 17 locally displayable 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 AP_MODE_TIMEOUT, specify 1-8 locally displayable type-A characters as a mode name.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

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

AP_INVALID_LU_NAME
The LU name specified in the *resource_name* parameter was not valid.

AP_INVALID_PARTNER_LU
The partner LU name specified in the *resource_name* parameter was not valid.

AP_INVALID_MODE_NAME
The mode name specified in the *resource_name* parameter was not valid.

AP_GLOBAL_TIMEOUT_NOT_DEFINED
The value AP_GLOBAL_TIMEOUT was specified for the *resource_type* parameter but there is no defined global timeout.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

3 つのルートを削除する

この verb は、 3270 エミュレーションまたは LUA (タイプ 0-3 の LU) に使用される LU を削除するために使用されます。

VCB 構造体

```
typedef struct delete_lu_0_to_3
{
     AP_UINT16           opcode;           /* verb operation code          */
     unsigned char       reserv2;          /* reserved                     */
     unsigned char       format;           /* reserved                     */
     AP_UINT16           primary_rc;       /* primary return code          */
     AP_UINT32           secondary_rc;     /* secondary return code        */
```

```
    unsigned char       lu_name[8];       /* LU name                    */
} DELETE_LU_0_TO_3;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DELETE_LU_0_TO_3

lu_name
Name of the local LU to be deleted. This is an 8-byte type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces if the name is shorter than 8 characters.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc

AP_INVALID_LU_NAME
The supplied LU name contained a character that was not valid.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc

ファイル名の変更
指定された LU 名は、 CS Linux システム上で定義された LU の名前ではありません。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **状態検査の追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

1 から 3 までの範囲の削除 (範囲)

この verb は、 3270 エミュレーションまたは LUA (タイプ 0-3 LU) に使用される LU の範囲を削除するために使用されます。

この verb に提供されるパラメーターには、 LU のベース名と NAU アドレスの範囲が含まれます。 削除される LU 名は、基本名と NAU アドレスを結合することによって判別されます。 For example, a base name of ルンメ combined with a NAU range of 11-14 would delete the LUs ランミ･ 011, ランミ 012, ルンメ 013, and ルメム 014.

指定された範囲内の名前を持つすべての LU が削除されます。範囲内の 1 つ以上の名前が存在しない場合、 CS Linux はエラーを戻しません。

VCB 構造体

```
typedef struct delete_lu_0_to_3_range
{
    AP_UINT16         opcode;            /* verb operation code             */
    unsigned char     reserv2;           /* reserved                        */
    unsigned char     format;            /* reserved                        */
    AP_UINT16         primary_rc;        /* primary return code             */
    AP_UINT32         secondary_rc;      /* secondary return code           */
    unsigned char     base_name[6];      /* Base name                       */
    unsigned char     min_nau;           /* Minimum NAU address in range    */
    unsigned char     max_nau;           /* Maximum NAU address in range    */
    unsigned char     name_attributes;   /* Extension type                  */
    unsigned char     base_number;       /* First extension number          */
    unsigned char     reserv5[16];       /* reserved                        */
} DELETE_LU_0_TO_3_RANGE;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
平均値が 1 から 3 までの値を 1 文字で削除

基本名
LU の名前のベース名。 これは、タイプ A の EBCDIC ストリング (文字で始まる) で、ベース名が 6 文字未満の場合は、右側に EBCDIC のスペースが埋め込まれます。 これは、 名前属性 パラメーターによって決定される 5 バイトまたは 6 バイトの長さにすることができます。 CS Linux は、各 NAU アドレスの 10 進値 (または 基本番号 パラメーターから始まる範囲内の数値) をこの名前に付加することによって、削除される LU の名前を判別します。

ミンナウ
1-255 の範囲内の最初の LU の NAU アドレス。

最大値
1-255 の範囲内の最後の LU の NAU アドレス。

名前属性
LU の拡張タイプを指定します。 可能な値は次のとおりです

追加なし
LU 名には、 NAU 番号に対応する番号があります。 数値は 10 進数で指定され、 基本名 パラメーターには 5 文字のみを含めることができます。

平均 BASE_NUMBER
基本番号 パラメーターに指定された値から、範囲内の LU の削除を開始します。

使用中の 16 進数名
LU 名の拡張は、 10 進数ではなく 16 進数で行われます。 この値を指定する場合、 基本名 パラメーターに 6 文字を含めることができます。

基本番号
名前属性 パラメーターに **平均** BASE_NUMBER が指定されている場合は、その範囲内の LU の削除を開始する番号を指定します。 この値は、 ミンナウ パラメーターの値の代わりに使用されます。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_NAU_ADDRESS
The *min_nau* or *max_nau* parameter was not valid.

AP_INVALID_LU_NAME
The *base_name* parameter contained a character that was not valid.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc

ファイル名の変更
指定された範囲内に名前を持つ LU が定義されていません。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **状態検査の追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

DELETE_LU_LU_PASSWORD

DELETE_LU_LU_PASSWORD deletes an LU-LU password associated with a local LU. LU-LU passwords are deleted automatically when the local LU is deleted; you need only use this verb if you need to remove the password but leave the LU configured.

VCB structure

```
typedef struct delete_lu_lu_password
{
    AP_UINT16        opcode;            /* verb operation code              */
    unsigned char    reserv2;           /* reserved                         */
    unsigned char    format;            /* reserved                         */
    AP_UINT16        primary_rc;        /* primary return code              */
    AP_UINT32        secondary_rc;      /* secondary return code            */
    unsigned char    lu_name[8];        /* LU name                          */
    unsigned char    lu_alias[8];       /* local LU alias                   */
    unsigned char    fqplu_name[17];    /* fully qualified partner LU name  */
    unsigned char    reserv3;           /* reserved                         */
} DELETE_LU_LU_PASSWORD;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
パスワードの追加を削除

lu_name
CS Linux に定義されている、ローカル LU の LU 名。 これは 8 バイトのタイプ A の EBCDIC ストリングで、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。 LU が LU 名の代わりに LU 別名で定義されていることを示すには、このパラメーターを 8 進ゼロに設定します。

lu_alias
CS Linux に定義されている、ローカル LU の LU 別名。 これは 8 バイトの ASCII ストリングで、ローカルで表示可能な文字を使用し、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。 *lu_name* がゼロに設定されている場合にのみ使用されます。

CP (デフォルト LU) に関連した LU を示すためには、 *lu_name* と *lu_alias* の両方を 2 進ゼロに設定します。

fqplu_name
CS Linux に対して定義されている、パートナー LU の完全修飾 LU 名。 この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc

ファイル名の変更
fqplu_name パラメーターが無効でした。

ファイル名の変更
lu_name パラメーターが無効でした。

エイプ無効ルリエイリアス
lu_alias パラメーターが無効でした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

プールの削除 (_L)

DELETE_LU_POOL は、以下のいずれかを行うために使用されます。

- 1 つ以上の LU をプールから除去する
- プールからすべての LU を除去し、プールを削除します。

この verb は、 LUs; を削除しません。定義されたままですが、どのプールにも関連付けられていません。

VCB 構造体

```
typedef struct delete_lu_pool
{
     AP_UINT16          opcode;             /* verb operation code          */
     unsigned char      reserv2;            /* reserved                     */
     unsigned char      format;             /* reserved                     */
     AP_UINT16          primary_rc;         /* primary return code          */
     AP_UINT32          secondary_rc;       /* secondary return code        */
     unsigned char      pool_name[8];       /* LU pool name                 */
     AP_UINT16          num_lus;            /* Number of specified LUs      */
     unsigned char      lu_names[10][8];    /* LU names                     */
} DELETE_LU_POOL;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DELETE_LU_POOL

pool_name
Name of the LU pool. This is an 8-byte EBCDIC string, padded on the right with spaces if the name is shorter than 8 characters.

num_lus
The number of LUs to be removed (the number of LU names in the *lu_names* list). The range is 1-10 when removing LUs from a pool without deleting it. To remove all LUs from the pool and delete the pool, specify zero.

lu_names
To remove one or more LUs from the pool without deleting the pool, specify the names of the LUs to be removed. The number of names specified must match the *num_lus* parameter. Each name is an 8-byte type A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces if the name is shorter than 8 characters.

If *num_lus* is set to zero, to remove all LUs from the pool and delete the pool, this parameter is not used.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_POOL_NAME
The supplied pool name was not valid.

AP_INVALID_LU_NAME
One or more of the specified LU names did not match the name of an LU in the pool.

AP_INVALID_NUM_LUS
The supplied *num_lus* parameter was not in the valid range.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

削除モード

DELETE_MODE は、モードの定義を削除します。 スナスフコマン および CPSVCMG などの SNA 定義のモードは削除できません。

VCB 構造体

```
typedef struct delete_mode
{
     AP_UINT16           opcode;           /* verb operation code          */
     unsigned char       reserv2;          /* reserved                     */
     unsigned char       format;           /* reserved                     */
     AP_UINT16           primary_rc;       /* primary return code          */
     AP_UINT32           secondary_rc;     /* secondary return code        */
     unsigned char       mode_name[8];     /* mode name                    */
} DELETE_MODE;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DELETE_MODE

mode_name
Name of the mode. This is an 8-byte type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces if the name is shorter than 8 characters.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_CP_OR_SNA_SVCMG_UNDELETABLE
The specified mode name is one of the SNA-defined mode names, and cannot be deleted.

AP_MODE_NAME_NOT_DEFD
The specified mode name is not the name of a mode defined on the CS Linux system.

AP_DEL_MODE_DEFAULT_SPCD
The specified mode was defined as the default mode using the DEFINE_DEFAULTS verb, so it cannot be deleted.

AP_MODE_UNDELETABLE
The specified mode name is one of the SNA-defined mode names, and cannot be deleted.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

パートナーの削除

DELETE_PARTNER_LU verb は、パートナー LU 定義を削除します。

VCB 構造体

```
typedef struct delete_partner_lu
{
    AP_UINT16         opcode;           /* verb operation code              */
    unsigned char     reserv2;          /* reserved                         */
    unsigned char     format;           /* reserved                         */
    AP_UINT16         primary_rc;       /* primary return code              */
    AP_UINT32         secondary_rc;     /* secondary return code            */
    unsigned char     fqplu_name[17];   /* fully qualified partner LU name  */
} DELETE_PARTNER_LU;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
追加の PARTNER_LU

fqplu_name
削除するパートナー LU の完全修飾 LU 名。 この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID、EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc

ファイル名の変更
指定された *fqplu_name* パラメーターが、定義済みパートナー LU 名と一致しませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

削除ポート

DELETE_PORT は、ポートを削除します。 この verb は、以下を削除します。

- ポートに関連付けられているすべてのリンク・ステーションおよび接続ネットワーク TG 。
- ポート上の LSs に関連付けられたすべての PU 、これらの PU が所有するすべての LU 、およびこれらの LU に関連するすべての LU-LU パスワード。

ポートは、 verb の発行時に非アクティブにする必要があります。

VCB 構造体

```
typedef struct delete_port
{
     AP_UINT16           opcode;           /* verb operation code          */
     unsigned char       reserv2;          /* reserved                     */
     unsigned char       format;           /* reserved                     */
     AP_UINT16           primary_rc;       /* primary return code          */
     AP_UINT32           secondary_rc;     /* secondary return code        */
     unsigned char       port_name[8];     /* name of port                 */
} DELETE_PORT;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DELETE_PORT

port_name
Name of port being deleted. This is an 8-byte ASCII string, right-padded with spaces if the name is shorter than 8 characters.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc

ポートフォリオ・ポート名
指定されたポート名は、 CS Linux システム上で定義されたポートの名前ではありませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc

エクスポート・ポートがアクティブ
指定されたポートは現在活動状態であるため、削除できません。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **状態検査の追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

アクセス権の削除 (_R)

DELETE_RCF_ACCESS は、以前に DEFINE_RCF_ACCESS を使用して指定された CS Linux リモート・コマンド機能 (RCF) へのアクセスを禁止します。 RCF の詳細については、「*IBM Communications Server for Linux* 管理ガイド上のデータ・センター・デプロイメント」を参照してください。

この verb は、 SPCF と UCF の両方へのアクセスを防止します そのうちの 1 つにアクセスできるようにするには、 DEFINE_RCF_ACCESS を使用してください。

この verb は、ドメイン構成ファイルに対して発行する必要があります。 CS Linux は、ノードの始動時に RCF アクセス・パラメーターに対して動作します。ノードの実行中に RCF アクセスが削除されると、ノードが停止して再始動されるまで、そのノードが稼働しているサーバーでは変更は有効になりません。

VCB structure

```
typedef struct delete_rcf_access
{
     AP_UINT16           opcode;           /* Verb operation code          */
     unsigned char       reserv2;          /* reserved                     */
     unsigned char       format;           /* reserved                     */
     AP_UINT16           primary_rc;       /* primary return code          */
     AP_UINT32           secondary_rc;     /* secondary return code        */
} DELETE_RCF_ACCESS;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
　追加の追加アクセス権

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
　AP_OK

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

セキュリティー・アクセス・リストの削除

DELETE_SECURITY_ACCESS_LIST は、以下のいずれかを行うために使用されます。

- セキュリティー・アクセス・リストを削除する。
- セキュリティー・アクセス・リストから 1 人以上のユーザーを削除しますが、構成されたリストはそのままに

そのユーザー名を使用してセットアップされたアクティブな 会話があるかどうかに関係なく、セキュリティー・アクセス・リストからユーザー名を削除することができます。 ユーザー名を削除してもアクティブ

会話には影響しませんが、呼び出し側プログラムは、削除されたユーザー名を使用してこれ以上の会話をセットアップすることはできません。

VCB structure

The DELETE_SECURITY_ACCESS_LIST verb contains a variable number of `security_user_name` structures; these define the user names to be deleted from the security access list. The user name structures are included at the end of the `delete_security_access_list` structure; the number of these structures is specified by the *num_users* parameter.

```
typedef struct delete_security_access_list
{
  AP_UINT16         opcode;                 /* verb operation code           */
  unsigned char     reserv2;                /* reserved                      */
  unsigned char     format;                 /* reserved                      */
  AP_UINT16         primary_rc;             /* primary return code           */
  AP_UINT32         secondary_rc;           /* secondary return code         */
  unsigned char     list_name[14];          /* name of this list             */
  unsigned char     reserv3[2];             /* reserved                      */
  AP_UINT32         num_users;              /* number of users to delete     */
} DELETE_SECURITY_ACCESS_LIST;
```

```
typedef struct security_user_name
{
  unsigned char     user_name[10];          /* user name to delete           */
} SECURITY_USER_NAME;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
セキュリティー・アクセス・リストの追加

リスト名
削除されるセキュリティー・アクセス・リストの名前、またはユーザー名が削除されるリストの名前。 これは、 1 から 14 文字の ASCII ストリングで、名前が 14 文字より短い場合は右側にスペースが埋め込まれます。これは、以前に定義されたセキュリティー・アクセス・リスト名と一致しなければなりません。

ユーザー数
セキュリティー・アクセス・リストから削除されるユーザー名の数。以下のとおりです。

- リストから 1 つ以上のユーザー名を削除するが、他のユーザー名を構成したままにするには、削除されるユーザー名の数を指定します。 これらはそれぞれ、以下で説明するように、ユーザー名構造によって定義される必要があります。
- セキュリティー・アクセス・リスト全体を削除するには、このパラメーターにゼロを指定し、ユーザー名は含めません。

削除するユーザー名ごとに、 ユーザー数で指定された数まで、以下のパラメーターを含む、DELETE_SECURITY_ACCESS_LIST 構造体の終わりに SECURITY_USER_NAME 構造体を追加します。

ユーザー名
削除されるユーザー名。 これは、このセキュリティー・アクセス・リストに現在定義されているユーザー名と一致する必要があります

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

ファイル・ リスト名の変更
指定されたセキュリティー・アクセス・リスト名がセキュリティー・アクセス・リスト名として定義されていません

ユーザー名の変更
指定されたユーザー名の 1 つ以上が、このセキュリティー・アクセス・リストに定義されたユーザーの名前と一致しませんでした。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

DELETE_TN3270_ACCESS

DELETE_TN3270_ACCESS is used to do one of the following:

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

VCB 構造体

```
typedef struct delete_tn3270_access
{
   AP_UINT16        opcode;              /* verb operation code             */
   unsigned char    reserv2;             /* reserved                        */
   unsigned char    format;              /* reserved                        */
   AP_UINT16        primary_rc;          /* primary return code             */
   AP_UINT32        secondary_rc;        /* secondary return code           */
   AP_UINT16        default_record;      /* is this the DEFAULT record?     */
   unsigned char    client_address[256]; /* address of TN3270 user          */
   AP_UINT32        num_sessions;        /* number of sessions to delete    */
   unsigned char    delete_options;      /* delete all sessions / delete    */
                                         /*  user?                          */
} DELETE_TN3270_ACCESS;
```

```
typedef struct tn3270_session_name
{
   AP_UINT16         port_number;        /* TCP/IP port num of session      */
                                         /*  to delete                      */
   unsigned char     listen_local_address[46];
                                         /* Local addr client connects to   */
} TN3270_SESSION_NAME;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DELETE_TN3270_ACCESS

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

AP_YES
This verb refers to the default TN3270 user record. The *client_address* parameter is reserved.

AP_NO
This verb refers to a normal TN3270 user record.

client_address
The TCP/IP address of the TN3270 user to be deleted, as specified on the DEFINE_TN3270_ACCESS verb. This is a null-terminated ASCII string, which 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`).

num_sessions
The number of sessions to be deleted, as follows:

- To delete one or more of the user's sessions but leave other sessions configured, specify the number of sessions that are being deleted. Each of these must be defined by its TCP/IP port number, as described below.
- To delete all sessions, or to delete the user, specify zero in this parameter and do not include any TCP/IP port numbers. Specify the type of deletion required in the *delete_options* parameter below.

delete_options
If the *num_sessions* parameter (see above) is nonzero, this parameter is ignored. If *num_sessions* is zero, specify one of the following values:

AP_ALL_SESSIONS
Delete all sessions but leave the TN3270 user configured.

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

For each session to be deleted, up to the number specified in *num_sessions*, append a TN3270_SESSION_NAME structure to the end of the DELETE_TN3270_ACCESS structure, containing the following parameters:

tn3270_session_name.port_number
The TCP/IP port number used for the session. This must match a port number defined for this TN3270 user.

tn3270_session_name.listen_local_address
The address on the local TN Server computer to which TN3270 clients connect.

- If this parameter was not specified when configuring the session, specify it as all binary zeros.
- If the address was specified when configuring the session, specify the same address in this parameter.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

ユーザー・アドレスの追加 （ _E ）
指定されたクライアント・アドレスが、どの TN3270 ユーザーに対して定義された TCP/IP アドレスと一致しませんでした

ファイル内のポート番号
指定された TCP/IP ポート番号は、このユーザーに定義されたどの TCP/IP ポート番号とも一致しませんでした

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

関連付けの削除 (_T)

DELETE_TN3270_ASSOCIATION は、ディスプレイ LU 名を指定した場合、ディスプレイ LU とプリンター LU の間の関連を削除します。

VCB 構造体

```
typedef struct delete_tn3270_association
{
   AP_UINT16        opcode;               /* verb operation code           */
   unsigned char    reserv2;              /* reserved                      */
   unsigned char    format;               /* reserved                      */
   AP_UINT16        primary_rc;           /* primary return code           */
   AP_UINT32        secondary_rc;         /* secondary return code         */
   unsigned char    display_lu_name[8];   /* Display LU name               */
} DELETE_TN3270_ASSOCIATION;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DELETE_TN3270_ASSOCIATION

display_lu_name
Specifies the name of the display LU whose association is to be deleted. This is an EBCDIC string padded on the right with EBCDIC spaces.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc

ファイル名の変更
表示 LU 名が有効な EBCDIC ストリングではありませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc

ファイル名の変更
指定されたディスプレイ LU に関連が定義されていません。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

DELETE_TN_REDIRECT

DELETE_TN_REDIRECT is used to delete a TN Redirector user, so that this user can no longer use TN Redirector to access a host.

VCB 構造体

```
typedef struct delete_tn_redirect
{
   AP_UINT16           opcode;              /* verb operation code             */
   unsigned char       reserv2;             /* reserved                        */
   unsigned char       format;              /* reserved                        */
   AP_UINT16           primary_rc;          /* primary return code             */
   AP_UINT32           secondary_rc;        /* secondary return code           */
   TN_REDIRECT_ADDRESS addr;                /* Uniquely defines record         */
} DELETE_TN_REDIRECT;
```

```
typedef struct tn_redirect_address
{
   AP_UINT16      default_record;      /* Is this the default record ?         */
   unsigned char  address_format;      /* IP address or fully-qualified name   */
   unsigned char  client_address[256]; /* Client address                       */
   AP_UINT16      port_number;         /* Port number that client connects on  */
   unsigned char  listen_local_address[46];  /* Local addr client connects to  */
   unsigned char  reserved[34];        /* reserved                             */
} TN_REDIRECT_ADDRESS;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DELETE_TN_REDIRECT

addr.default_record
Specifies whether this verb refers to the default TN Redirector user record that is used by any TN Redirector user not explicitly identified by a TCP/IP address (deleting this record means that such users cannot access TN Redirector). Possible values are:

AP_YES
This verb refers to a default record. The *client_address* and *address_format* parameters are reserved.

AP_NO
This verb refers to a normal TN Redirector user record.

addr.address_format
Specifies the format of the *client_address* parameter. Possible values are:

AP_ADDRESS_IP
IP address (either IPv4 or IPv6)

AP_ADDRESS_FQN
Alias or fully qualified name

addr.client_address
The TCP/IP address of the computer on which the Telnet client runs. This is a null-terminated ASCII string, which 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`).

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).

addr.port_number
The number of the server TCP/IP port that the Telnet client uses to access the TN server node.

addr.listen_local_address
The address on the local TN Server computer to which TN3270 clients connect.

- If this parameter was not specified when configuring the TN redirection record, specify it as all binary zeros.
- If the address was specified when configuring the TN redirection record, specify the same address in this parameter.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc

AP_INVALID_CLIENT_ADDRESS
The specified addressing information did not match any defined TN Redirector user.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

DELETE_TP

DELETE_TP deletes a TP definition.

VCB structure

```
typedef struct delete_tp
{
   AP_UINT16        opcode;                 /* verb operation code         */
   unsigned char    reserv2;                /* reserved                    */
   unsigned char    format;                 /* reserved                    */
   AP_UINT16        primary_rc;             /* primary return code         */
   AP_UINT32        secondary_rc;           /* secondary return code       */
   unsigned char    tp_name[64];            /* TP name                     */
} DELETE_TP;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
追加の削除 （ T ）

Tp_name
削除する TP の名前。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

ファイル名の変更
Tp_name パラメーターが、定義済みの TP の名前と一致しませんでした。

SYSTEM_TP_CANT_BE_DELETED が削除されています
指定された TP 名は、 CS Linux によって内部で使用される TP の名前であり、削除することはできません。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

DELETE_TP_LOAD_INFO

The DELETE_TP_LOAD_INFO verb deletes a TP load information entry.

VCB 構造体

```
typedef struct delete_tp_load_info
{
   AP_UINT16          opcode;                /* verb operation code        */
   unsigned char      reserv2;               /* reserved                   */
   unsigned char      format;                /* reserved                   */
   AP_UINT16          primary_rc;            /* primary return code        */
   AP_UINT32          secondary_rc;          /* secondary return code      */
   unsigned char      tp_name[64];           /* TP name                    */
   unsigned char      lu_alias[8];           /* LU alias                   */
} DELETE_TP_LOAD_INFO;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DELETE_TP_LOAD_INFO

tp_name
The TP name of the TP load info entry to be deleted. This is a 64-byte EBCDIC string, padded on the right with spaces if the name is shorter than 64 characters.

lu_alias
The LU alias of the TP load info entry to be deleted. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes.

This parameter can be used only if the TP is an APPC application; it is reserved if the TP is a CPI-C application.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_TP_NAME
The *tp_name* parameter did not match the name of a defined TP.

AP_INVALID_LU_ALIAS
The *lu_alias* parameter did not match any defined LU alias specified for a TP load info entry for the TP name specified.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

DELETE_USERID_PASSWORD

DELETE_USERID_PASSWORD deletes a password associated with a user ID, or removes profiles for a user ID and password.

VCB 構造体

```
typedef struct delete_userid_password
{
   AP_UINT16               opcode;             /* verb operation code        */
   unsigned char           reserv2;            /* reserved                   */
   unsigned char           format;             /* reserved                   */
   AP_UINT16               primary_rc;         /* primary return code        */
   AP_UINT32               secondary_rc;       /* secondary return code      */
   AP_UINT16               delete_type;        /* type of delete             */
   unsigned char           user_id[10];        /* user id                    */
   USERID_PASSWORD_CHARS   password_chars;     /* password characteristics   */
} DELETE_USERID_PASSWORD;
```

```
typedef struct userid_password_chars
{
   unsigned char           description[32];    /* resource description       */
   unsigned char           reserv2[16];        /* reserved                   */
   AP_UINT16               profile_count;      /* number of profiles         */
   AP_UINT16               reserv1;            /* reserved                   */
   unsigned char           password[10];       /* password                   */
   unsigned char           profiles[10][10];   /* profiles                   */
} USERID_PASSWORD_CHARS;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_DELETE_USERID_PASSWORD

delete_type
Specifies how this verb is being used. Possible values are:

AP_REMOVE_USER
Delete the user, password, and all associated profiles.

AP_REMOVE_PROFILES
Delete the specified profiles.

user_id
User identifier. This is a 10-byte type-AE EBCDIC character string, padded on the right with EBCDIC spaces if the name is shorter than 10 characters.

password_chars.description
This parameter is ignored.

password_chars.profile_count
Number of profiles to be deleted. If *delete_type* is set to AP_REMOVE_USER, this parameter is reserved.

password_chars.password
This parameter is ignored.

password_chars.profiles
Profiles associated with user. Each of these is a 10-byte type-AE EBCDIC character string, padded on the right with EBCDIC spaces if the profile name is shorter than 10 characters.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

個のプロファイルの追加
削除のタイプ パラメーターが **モデル・プロファイル** （ _C ）に設定されましたが、プロファイルが指定されていませんでした。

認識されていないユーザー
ユーザー *ID* パラメーターが、定義されたユーザー ID と一致しませんでした。

ファイルの追加の更新のタイプ
削除のタイプ パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

ノードの切断

アプリケーションは、この verb を使用して、ノードへの NOF verb の発行を終了したときに、そのハンドルを CS Linux ノードに解放します。 アプリケーションが切断を希望するノードは、呼び出しの *target_handle* パラメーターによって識別されます。 verb が正常に完了すると、ノードを識別するターゲット・ハンドルが無効になります。

アプリケーションは、オープン・ノードが終了する前に、常に DISCONNECT_NODE を発行して、 CS Linux がアプリケーションに関連したリソースを解放できるようにする必要があります。

この verb は、実行中のノードのターゲット・ハンドルを解放するため、またはノードが稼働していないサーバーのターゲット・ハンドルを解放するために発行できます。

VCB structure

```
typedef struct disconnect_node
{
     AP_UINT16           opcode;           /* Verb operation code          */
     unsigned char       reserv2;          /* reserved                     */
     unsigned char       format;           /* reserved                     */
     AP_UINT16           primary_rc;       /* Primary return code          */
     AP_UINT32           secondary_rc;     /* Secondary return code        */
} DISCONNECT_NODE;
```

提供されるパラメーター

オペコード
接続ノードの接続解除

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

secondary_rc
Not used.

戻りパラメーター: 状態チェック

状態チェックのために verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc

AP_VERB_IN_PROGRESS
指定されたターゲット・ハンドルは、このハンドルに対して発行された直前の verb が未処理のため、解放できません ノードからの切断を試行する前に、ターゲット・ハンドルのすべての verb を完了する必要があります。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **状態検査の追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

初期ノード

この verb は、以前に定義したノードを開始します アプリケーションはまず、ノードのターゲット・ハンドルを取得するために CONNECT_NODE を発行する必要があります。その後、 INIT_NODE 呼び出しでこのターゲット・ハンドルを使用して、開始するノードを識別します。

この verb は、ノードが実行されていないサーバーに対して発行する必要があります。

VCB structure

```
typedef struct init_node
{
     AP_UINT16           opcode;             /* verb operation code          */
     unsigned char       reserv2;            /* reserved                     */
     unsigned char       format;             /* reserved                     */
     AP_UINT16           primary_rc;         /* primary return code          */
     AP_UINT32           secondary_rc;       /* secondary return code        */
} INIT_NODE;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
AP_INIT_NODE

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

secondary_rc
未使用。

Returned parameters: parameter check

If the verb does not execute because of a parameter check, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_NODE_NAME
The node name specified in the configuration file does not match the name of the CS Linux computer to which the verb was issued.

AP_NOT_SERVER
The node name specified in the configuration file matches the name of the CS Linux computer, but the specified computer is a client (not a server) and cannot run the node.

AP_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.

Returned parameters: state check

If the verb does not execute because of a state check, CS Linux returns the following parameters.

primary_rc
AP_STATE_CHECK

secondary_rc
Possible values are:

AP_NODE_ALREADY_STARTED
The specified node has already been started.

AP_RESOURCE_NOT_LOADED
The node was not started because CS Linux detected one or more errors while attempting to load its configuration. Check the error log file for messages giving more details of the errors.

AP_INVALID_VERSION
The node was not started because there was a version mismatch between components of the CS Linux software. If you have upgraded your CS Linux license to include additional functions or users, check that you are using the correct version of the licensing software.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_STATE_CHECK, which are common to all NOF verbs.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

INITIALIZE_SESSION_LIMIT

The INITIALIZE_SESSION_LIMIT verb initializes the session limits for a combination of local LU, partner LU, and mode.

You must issue this verb before you issue an ACTIVATE_SESSION verb.

This verb can be issued from a NOF application running on a client. If it runs on an AIX or Linux client, the NOF application must run with the userid `root`, or with a userid that is a member of the `sys` group (AIX) or `sna` group (Linux).

VCB structure

```
typedef struct initialize_session_limit
{
   AP_UINT16        opcode;                 /* verb operation code         */
   unsigned char    reserv2;                /* reserved                    */
   unsigned char    format;                 /* reserved                    */
   AP_UINT16        primary_rc;             /* primary return code         */
   AP_UINT32        secondary_rc;           /* secondary return code       */
   unsigned char    lu_name[8];             /* local LU name               */
   unsigned char    lu_alias[8];            /* local LU alias              */
   unsigned char    plu_alias[8];           /* partner                     */
   unsigned char    fqplu_name[17];         /* fully qualified partner     */
                                            /*  LU name                    */
   unsigned char    reserv3;                /* reserved                    */
   unsigned char    mode_name[8];           /* mode name                   */
   unsigned char    reserv3a;               /* reserved                    */
   unsigned char    set_negotiable;         /* set max negotiable limit?   */
   AP_UINT16        plu_mode_session_limit; /* session limit               */
   AP_UINT16        min_conwinners_source;  /* minimum source contention   */
                                            /*  winner sessions            */
   AP_UINT16        min_conwinners_target;  /* minimum target contention   */
                                            /*  winner sessions            */
   AP_UINT16        auto_act;               /* auto activation limit       */
   unsigned char    reserv4[4];             /* reserved                    */
   AP_UINT32        sense_data;             /* sense data                  */
} INITIALIZE_SESSION_LIMIT;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
追加の初期セッション制限

lu_name
CS Linux に定義されている、ローカル LU の LU 名。 これは 8 バイトのタイプ A の EBCDIC ストリングで、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。 LU が LU 名の代わりに LU 別名で定義されていることを示すには、このパラメーターを 8 進ゼロに設定します。

lu_alias
CS Linux に定義されている、ローカル LU の LU 別名。 これは 8 バイトの ASCII ストリングで、ローカルで表示可能な文字を使用し、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。 *lu_name* がゼロに設定されている場合にのみ使用されます。

CP (デフォルト LU) に関連した LU を示すためには、 *lu_name* と *lu_alias* の両方を 2 進ゼロに設定します。

plu_alias
パートナー LU の LU 別名。 これは 8 バイトの ASCII ストリングで、ローカルで表示可能な文字を使用し、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。 パートナー LU が LU 別名の代わりに完全修飾 LU 名によって定義されていることを示すには、このパラメーターを 8 桁の 2 進ゼロに設定します。

fqplu_name
CS Linux に対して定義されている、パートナー LU の完全修飾 LU 名。 このパラメーターは、*plu_alias* フィールドがゼロに設定されている場合のみ使用されます。 *plu_alias* が指定されている場合は無視されます。

この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

モード名
LU によって使用されるモードの名前。 これは 8 バイトの英数字のタイプ A の EBCDIC ストリング (文字で始まる) で、名前が 8 バイトより短い場合は、右側に EBCDIC のスペースが埋め込まれます。

セット折衝可能
DEFINE_MODE で定義されている、このモードの最大折衝可能セッション限度を変更するかどうかを指定します。 可能な値は次のとおりです

類人猿
この LU - LU モードの組み合わせの折衝可能な最大セッション限度として、 *plu_mode_session_limit* によって指定された値を使用してください。

アプ・ノー
ネゴシエーション可能な最大セッション限度は、モードに指定されている値のままにします。

plu_mode_session_limit
この LU - LU モードの組み合わせに対する要求された合計セッション限度。このモードを使用する 2 つの LU 間で許可される並列セッションの最大数。 1-32,767 の範囲内の値を指定します (これは、DEFINE_LOCAL_LU verb のローカル LU に指定されたセッション限度を超えてはなりません) 。 この値は、パートナー LU と折衝することができます。

***min_conwinners_source* ソース**
ローカル LU がコンテンション勝者であるこのモードを使用するセッションの最小数。 0-32,767 の範囲の値を指定してください。 *min_conwinners_source* ソース パラメーターと *min_conwinners_target* パラメーターの合計は、 *plu_mode_session_limit* パラメーターを超えてはなりません。

min_conwinners_target
パートナー LU がコンテンション勝者であるこのモードを使用するセッションの最小数。 0-32,767 の範囲の値を指定してください。 *min_conwinners_source* ソース パラメーターと *min_conwinners_target* パラメーターの合計は、 *plu_mode_session_limit* パラメーターを超えてはなりません。

自動実行
自動的に活動化するセッションの数。 0-32,767 の範囲の値を指定します (これは、DEFINE_LOCAL_LU verb でローカル LU に指定された *plu_mode_session_limit* パラメーターまたはセッション限度を超えてはなりません) 。 自動的に活動化されるセッションの実際の数は、この値の最小値と、ローカル LU のコンテンション勝者セッションの折衝済み最小数となります。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

secondary_rc
可能な値は次のとおりです

折衝済みのアプ
セッション限度は初期設定されましたが、 1 つ以上の値がパートナー LU によって折衝されました。

指定されたアプ
セッション限度は、パートナー LU によって折衝されることなく、要求通りに初期設定されました。

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_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.

AP_CANT_CHANGE_TO_ZERO
The *plu_mode_session_limit* parameter cannot be set to zero using this verb; use RESET_SESSION_LIMIT instead.

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

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

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

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

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: state check

If the verb does not execute because of a state error, CS Linux returns the following parameters.

primary_rc
AP_STATE_CHECK

secondary_rc

AP_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.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_STATE_CHECK, which are common to all NOF verbs.

Returned parameters: session allocation error

If the verb does not execute because of a session allocation error, CS Linux returns the following parameters:

primary_rc
AP_ALLOCATION_ERROR

secondary_rc

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

sense_data
The SNA sense data associated with the allocation failure.

Returned parameters: CNOS processing errors

If the verb does not execute because of an error, CS Linux returns the following parameters.

primary_rc

AP_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 CS Linux log file for information about the error condition, and correct it before retrying this verb.

primary_rc
AP_CNOS_PARTNER_LU_REJECT

secondary_rc

AP_CNOS_COMMAND_RACE_REJECT
The verb failed because the specified mode was being accessed by another administration program (or internally by the CS Linux software) for session activation or deactivation, or for session limit processing. The application should retry the verb, preferably after a timeout to allow the race condition to be cleared.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

オープン・ファイル

アプリケーションは、ドメイン・リソースを管理するため、または CS Linux LAN 上のバックアップ・サーバーを管理するための **スネネット** ファイル。 にアクセスするために、この verb を使用して CS Linux ドメイン構成ファイルにアクセスします。

この verb は、ヌルのターゲット・ハンドルで発行する必要があります これが正常に完了すると、 CS Linux はファイルを識別するハンドルを戻します。このハンドルは、アプリケーションが他の NOF verb で使用して、 verb のターゲットを指示することができます。

VCB 構造体

```
typedef struct open_file
{
    AP_UINT16        opcode;              /* verb operation code            */
    unsigned char    reserv2;             /* reserved                       */
    unsigned char    format;              /* reserved                       */
    AP_UINT16        primary_rc;          /* primary return code            */
    AP_UINT32        secondary_rc;        /* secondary return code          */
    CONFIG_FILE      file_info;           /* definition of file requested   */
    AP_UINT32        target_handle;       /* handle for subsequent verbs    */
    unsigned char    reserv3[4];          /* reserved                       */
} OPEN_FILE;
```

```
typedef struct config_file
{
    unsigned char    requested_role;      /* config file requested          */
    unsigned char    role_supplied;       /* config file returned           */
    unsigned char    system_name[128];    /* computer name where file       */
                                          /*  located                       */
    unsigned char    file_name[81];       /* file name                      */
} CONFIG_FILE;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
ファイルの追加 （ _C ）

file_info.requested_role
オープンするファイルのタイプ。 可能な値は次のとおりです

AP_コントローラー
ドメイン構成ファイルの制御コピーを開きます。 この値は、アプリケーションがドメイン・リソースの構成を変更する verb を発行しようとする場合に使用する必要があります。

AP_BACKUP
使用可能であれば、ドメイン構成ファイルの制御コピーを開きます。それ以外の場合は、バックアップ・コピーです。 この値は、アプリケーションが QUERY_* verb のみを発行する場合に使用

できます。構成を変更する必要がある場合は、 AP_コントローラーを使用する必要があります。これは、バックアップ構成ファイルへの書き込みアクセス権を取得できないためです。

ネット・スネックネット
コントローラー・サーバー上の **スネネット** ファイル。 を開く

追加情報 LOAD_INFO
トランザクション・プログラム (TP)のロード方法に関する情報が含まれている、ローカル・マシン上のファイルへの接続を開きます。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

target_handle
このファイルに送信される後続の verb で使用するための戻り値。

提供された *file_info.role_*供給
要求さ要求のロール が AP_BACKUP に設定されている場合、このパラメーターは、制御構成ファイルまたはバックアップ・ファイル用のファイル・ハンドルが戻されるかどうかを示します。 可能な値は次のとおりです

AP_コントローラー
構成ファイルの制御。

AP_BACKUP
構成ファイルのバックアップ。

要求さ要求のロールのその他のすべての値については、このパラメーターは未定義です。

ファイル・デバイス・システム名
ファイルが配置されている CS Linux コンピューターの名前。

ファイル・ *info.file_name*
ファイルの名前。 このパラメーターは、 1 文字から 80 文字の ASCII ストリングで、その後にヌル (0x00) 文字が続きます。

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

追加ファイル・ファイル名
ファイル名 パラメーターに、有効な構成ファイル名が指定されていませんでした。

ファイル・ファイルの追加情報
ファイル情報 構造体内のパラメーターの 1 つが無効でした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc
可能な値は次のとおりです

接続の追加が行われない
CS Linux は、ファイルへのローカル通信パスをセットアップできませんでした。

AP_FILE_BAD_RECORD
CS Linux が構成ファイルにエラーを検出しました。 エラー・ログ・ファイルで、エラーの詳細を示すメッセージを確認してください。

ファイル・ファイルの追加が使用不可
アプリケーションが制御またはバックアップ構成ファイル、または **スネネット** ファイルを要求しましたが、使用可能なコントローラーまたはバックアップ・サーバーがありませんでした。 これは通常、新規サーバーがコントローラーとしてテークオーバーしているときに発生する一時的な状態です。

アプリケーションがサーバー指示を受信するように登録されている場合は、これらの指示の 旗 パラメーターを検査して、新しいサーバーがいつコントローラーとして正常に引き継がれたかを判別し、 OPEN_FILE verb を再試行することができます。 詳しくは、 648 ページの『SERVER_INDICATION』を参照してください。 あるいは、成功するまでインターバルで OPEN_FILE を単に再試行することもできます。

バージョンの追加
構成ファイル・ヘッダー内の CS Linux バージョン番号が、使用している CS Linux ソフトウェアのバージョンと一致していません。 正しいファイルを持っていることを確認してください。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **状態検査の追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

PATH_SWITCH

PATH_SWITCH は、 CS Linux が現在アクティブな高速トランスポート・プロトコル (RTP) 接続を別のパスに切り替えることを要求します。 CS Linux が適切なパスを検出できない場合は、接続は変更されません。

VCB 構造体

```
typedef struct path_switch
{
    AP_UINT16        opcode;                /* verb operation code          */
    unsigned char    reserv2;               /* reserved                     */
    unsigned char    format;                /* reserved                     */
    AP_UINT16        primary_rc;            /* primary return code          */
    AP_UINT32        secondary_rc;          /* secondary return code        */
    unsigned char    rtp_connection_name[8]; /* RTP connection name         */
} PATH_SWITCH;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_PATH_SWITCH

rtp_connection_name
The RTP connection for which a change in path is requested. This is an 8-byte string in a locally displayable character set. All eight bytes are significant and must be set.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc

AP_INVALID_RTP_CONNECTION
The value specified for the *rtp_connection_name* parameter did not match the name of an existing RTP connection.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: state check

If the verb does not execute because of a state error, CS Linux returns the following parameters.

primary_rc
AP_STATE_CHECK

secondary_rc

AP_PATH_SWITCH_IN_PROGRESS
CS Linux is currently changing the path for the RTP connection specified by the *rtp_connection_name* parameter.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_STATE_CHECK, which are common to all NOF verbs.

Returned parameters: path switch disabled

If the verb does not execute because the RTP partner node has disabled path switch by setting the path switch timer to zero, CS Linux returns the following parameter:

primary_rc
AP_PATH_SWITCH_DISABLED

secondary_rc
(No secondary return code is returned.)

Returned parameters: path switch failure

If the verb does not execute because the path switch attempt fails, CS Linux returns the following parameter:

primary_rc
AP_UNSUCCESSFUL

secondary_rc
(No secondary return code is returned.)

Returned parameters: node check

If the verb does not execute because the system has not been built with RTP support, CS Linux returns the following parameter:

primary_rc
AP_INVALID_VERB

secondary_rc
(No secondary return code is returned.)

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

照会アクティブ・トランザクション

QUERY_ACTIVE_TRANSACTION は、 CS Linux 管理サービス・コンポーネントに認識されているアクティブな複数ドメイン・サポート (MDS) トランザクションに関する情報を戻します。 アクティブ・トランザクションは、応答がまだ受信されていない MDS 要求です。

この verb は、使用されるオプションに応じて、単一のトランザクション、または複数のトランザクションに関する情報を取得するために使用できます。

この verb は実行中のノードに対して発行する必要があります

VCB 構造体

```
typedef struct query_active_transaction
{
   AP_UINT16       opcode;                 /* verb operation code           */
   unsigned char   reserv2;                /* reserved                      */
   unsigned char   format;                 /* reserved                      */
   AP_UINT16       primary_rc;             /* primary return code           */
   AP_UINT32       secondary_rc;           /* secondary return code         */
   unsigned char   *buf_ptr;               /* pointer to buffer             */
   AP_UINT32       buf_size;               /* buffer size                   */
   AP_UINT32       total_buf_size;         /* total buffer size required    */
   AP_UINT16       num_entries;            /* number of entries             */
   AP_UINT16       total_num_entries;      /* total number of entries       */
   unsigned char   list_options;           /* listing options               */
   unsigned char   reserv3;                /* reserved                      */
   unsigned char   fq_req_loc_cp_name[17]; /* fq cp name of transaction     */
                                           /*  requestor                    */
   unsigned char   req_agent_appl_name[8]; /* appl name of transaction      */
                                           /*  requestor                    */
   unsigned char   seq_num_dt[17];         /* sequence number date/time     */
} QUERY_ACTIVE_TRANSACTION;
```

```
typedef struct active_transaction_data
{
   AP_UINT16       overlay_size;           /* size of returned entry        */
   unsigned char   fq_origin_cp_name[17];  /* cp name of transaction origin */
   unsigned char   origin_ms_appl_name[8]; /* appl name of transaction      */
                                           /*  origin                       */
   unsigned char   fq_dest_cp_name[17];    /* cp name of transaction        */
                                           /*  destination                  */
   unsigned char   dest_ms_appl_name[8];   /* appl name of transaction dest */
   unsigned char   fq_req_loc_cp_name[17]; /* fq cp name of transaction     */
                                           /*  requestor                    */
   unsigned char   req_agent_appl_name[8]; /* appl name of transaction      */
                                           /*  requestor                    */
   unsigned char   seq_num_dt[17];         /* sequence number date/time     */
   unsigned char   reserva[20];            /* reserved                      */
} ACTIVE_TRANSACTION_DATA;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
照会アクティブ・トランザクション

buf_ptr
CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

num_entries
データが戻されるトランザクションの最大数。 特定の範囲ではなく、特定のトランザクションのデータを要求するには、 1 という値を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、 CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータの戻りを開始する必要があるリスト内の位置。 可能な値は次のとおりです

リストの最初のリスト （ _R ）
リストの最初の項目から開始します。

リストを含む （ 包括的 ）
fq_req_loc_cp_name、 *req_agent_appl_name*、および *seq_num_dt* パラメーターによって指定されたエントリーから開始します。

次への AP_LIST_FROM_NEXT
fq_req_loc_cp_name、 *req_agent_appl_name*、および *seq_num_dt* パラメーターで指定されたエントリーの直後のエントリーから開始します。

このリストは、 *fq_req_loc_cp_name* の順に並べられ、次に *req_agent_appl_name seq_num_dt* の順に番号順に並べられています。 リストの順序とアプリケーションでの特定のエントリーの取得方法について詳しくは、 34 ページの『List options for QUERY_* Verbs』を参照してください。

fq_req_loc_cp_name
トランザクション要求側の完全修飾制御点名。 *list_options* を **リストの最初のリスト （ _R ）**に設定すると、このパラメーターは無視されます。 この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

req_agent_appl_name
トランザクション要求側のアプリケーション名。 *list_options* を **リストの最初のリスト （ _R ）**に設定すると、このパラメーターは無視されます。

この名前は、通常、タイプ 1134 文字 (大文字の A から Z および数字 0-9) を使用して EBCDIC ストリングです。あるいは、 *SNA* 管理サービス解説書で指定された MS 規律固有アプリケーション・プログラムの 1 つにすることもできます。 ストリングは 8 文字の長さでなければなりません。必要に応じて、右側に EBCDIC スペース文字 (0x40 **バイト**) を埋め込む必要があります。

seq_num_dt
IBM SNA 形式マニュアルで定義されている、元のトランザクションのシーケンス 番号日付/時刻相関関係子 (17 バイトの長さ) 。 *list_options* を **リストの最初のリスト （ _R ）**に設定すると、このパラメーターは無視されます。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

buf_size
提供されたバッファーに戻された情報の長さ。

total_buf_size
要求されたすべてのリスト情報を戻すために必要となるバッファーのサイズを示す戻り値。 *buf_size* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

num_entries
データ・バッファーに戻されたエントリーの数。

total_num_entries
使用可能な項目の合計数。 *num_entries* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

データ・バッファー内の各項目は、以下のパラメーターで構成されます。

アクティブ・トランザクション・データ.オーバーレイ・サイズ
戻された **アクティブ・トランザクション・データ** 構造体のサイズ。すなわち、データ・バッファー内の次のエントリーの先頭までのオフセット。

アプリケーションが戻されたバッファーを調べて、各 **アクティブ・トランザクション・データ** 構造体を順番に検出する必要がある場合は、この値を使用して次のデータ構造の正しいオフセットに移動しなければなりません。また、 C `sizeof()` 演算子を使用してはなりません。 これは、返されるオーバーレイのサイズが、 CS Linux の将来のリリースで増加する可能性があるためです。戻されたオーバーレイ・サイズを使用すると、アプリケーションが今後のリリースで引き続き動作することが保証されます。

active_transaction_data.fq_origin_cp_name
トランザクションの起点の完全修飾制御点名。 この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

アクティブ・トランザクション・データ *. origin_ms_appl_name*
トランザクションの起点のアプリケーション名。 この名前は、通常、タイプ 1134 文字 (大文字の A から Z および数字 0-9) を使用して 8 文字の EBCDIC ストリングです。あるいは、 システム・ネットワーク体系*:* 管理サービス参照 で指定された MS 規律固有アプリケーション・プログラムの 1 つ (「参考文献」を参照) のいずれかになります。

アクティブ・トランザクション・データ *.fq_dest_cp_name*
トランザクションの宛先の完全修飾制御点名。 この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

アクティブ・トランザクション・データ *. dest_ms_appl_name*
トランザクションの宛先アプリケーションのアプリケーション名。 この名前は、通常、タイプ 1134 文字 (大文字の A から Z および数字 0-9) を使用して 8 文字の EBCDIC ストリングです。あるいは、 システム・ネットワーク体系*:* 管理サービス参照 で指定された MS 規律固有アプリケーション・プログラムの 1 つ (「参考文献」を参照) のいずれかになります。

アクティブ・トランザクション・データ *.fq_req_loc_cp_name*
トランザクション要求側の完全修飾制御点名。 この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

アクティブ・トランザクション・データ *.req_agent_appl_name*
トランザクション要求側のアプリケーション名。 この名前は、通常、タイプ 1134 文字 (大文字の A から Z および数字 0-9) を使用して 8 文字の EBCDIC ストリングです。あるいは、 システム・ネットワーク体系*:* 管理サービス参照 で指定された MS 規律固有アプリケーション・プログラムの 1 つ (「参考文献」を参照) のいずれかになります。

アクティブ・トランザクション・データ *.seq_num_dt*
IBM SNA 形式マニュアルで定義されている、元のトランザクションのシーケンス 番号日付/時刻相関関係子 (17 バイトの長さ) 。

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

非アクティブ・アクティブ・トランザクション
制御点名、アプリケーション名、またはシーケンス番号の相関関係子が、アクティブ・トランザクションのものと一致しませんでした。

ファイルの追加リスト・オプション
list_options パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、**パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: function not supported

If the verb does not execute successfully because the local node configuration does not support it, CS Linux returns the following parameters:

primary_rc

AP_FUNCTION_NOT_SUPPORTED
The local node does not support MS network management functions; this is defined by the *mds_supported* parameter on the DEFINE_NODE verb.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

QUERY_ADJACENT_NN

The QUERY_ADJACENT_NN verb returns information about adjacent network nodes (the network nodes to which CP-CP sessions are active or have been active at some time). It can be used only if the CS Linux node is a network node, and is not valid if it is an end node or LEN node.

This verb can be used to obtain information about a specific adjacent network node, or about multiple adjacent network nodes, depending on the options used.

This verb must be issued to a running node.

VCB 構造体

```
typedef struct query_adjacent_nn
{
   AP_UINT16        opcode;              /* verb operation code              */
   unsigned char    reserv2;             /* reserved                         */
   unsigned char    format;              /* reserved                         */
   AP_UINT16        primary_rc;          /* primary return code              */
   AP_UINT32        secondary_rc;        /* secondary return code            */
   unsigned char    *buf_ptr;            /* pointer to buffer                */
   AP_UINT32        buf_size;            /* buffer size                      */
   AP_UINT32        total_buf_size;      /* total buffer size required       */
   AP_UINT16        num_entries;         /* number of entries                */
   AP_UINT16        total_num_entries;   /* total number of entries          */
   unsigned char    list_options;        /* listing options                  */
   unsigned char    reserv3;             /* reserved                         */
   unsigned char    adj_nncp_name[17];   /* CP name of adjacent Network Node */
} QUERY_ADJACENT_NN;
```

```
typedef struct adj_nncp_data
{
   AP_UINT16        overlay_size;        /* size of returned entry           */
   unsigned char    adj_nncp_name[17];   /* CP name of adjacent network node */
   unsigned char    cp_cp_sess_status;   /* CP-CP session status             */
   AP_UINT32        out_of_seq_tdus;     /* out of sequence TDUs             */
   AP_UINT32        last_frsn_sent;      /* last FRSN sent                   */
   AP_UINT32        last_frsn_rcvd;      /* last FRSN received               */
```

```
   unsigned char    reserva[20];         /* reserved                          */
} ADJ_NNCP_DATA;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_ADJACENT_NN

buf_ptr
A pointer to a data buffer that CS Linux will use to return the requested information.

buf_size
Size of the supplied data buffer.

num_entries
Maximum number of adjacent NNs for which data should be returned. To request data for a specific adjacent NN rather than a range, specify the value 1. To return as many entries as possible, specify zero; in this case, CS Linux will return the maximum number of entries that can be accommodated in the supplied data buffer.

list_options
The position in the list of adjacent NNs from which CS Linux should begin to return data. Possible values are:

AP_FIRST_IN_LIST
Start at the first entry in the list.

AP_LIST_INCLUSIVE
Start at the entry specified by the *adj_nncp_name* parameter.

AP_LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *adj_nncp_name* parameter.

For more information about how the list is ordered and how the application can obtain specific entries from it, see “List options for QUERY_* Verbs” on page 34.

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 AP_FIRST_IN_LIST.

The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

adj_nncp_data.overlay_size
The size of the returned `adj_nncp_data` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `adj_nncp_data` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

adj_nncp_data.adj_nncp_name
Fully qualified name of the adjacent NN. The name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of 1-8 A-string characters, an EBCDIC dot (period) character, and a network name of 1-8 A-string characters.

adj_nncp_data.cp_cp_sess_status
Status of the CP-CP session to the adjacent NN. Possible values are:

AP_ACTIVE
The session is active.

AP_CONWINNER_ACTIVE
The session (a contention-winner session) is active.

AP_CONLOSER_ACTIVE
The session (a contention-loser session) is active.

AP_INACTIVE
The session is inactive.

adj_nncp_data.out_of_seq_tdus
Number of out-of-sequence TDUs received from this node.

adj_nncp_data.last_frsn_sent
The last Flow Reduction Sequence Number (FRSN) sent to this node.

adj_nncp_data.last_frsn_rcvd
The last Flow Reduction Sequence Number (FRSN) received from this node.

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_ADJ_NNCP_NAME
The *list_options* parameter was set to AP_LIST_INCLUSIVE, to list all entries starting from the supplied name, but the *adj_nncp_name* parameter was not valid.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: function not supported

If the verb does not execute successfully because the local node is not a network node, CS Linux returns the following parameters:

primary_rc

AP_FUNCTION_NOT_SUPPORTED
The local node is an end node or LEN node. This verb is valid only at a network node.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

照会可用性 _TP

QUERY_AVAILABLE_TP は、アクティブな呼び出し可能 TP (RECEIVE_ALLOCATE verb を発行した APPC アプリケーション、または Accept_Conversation コールまたは Accept_Incoming コールを発行した CPI-C アプリケーション) に関する情報を戻します。 この verb は、使用されるオプションに応じて、特定の TP に関する情報、または複数の TP に関する情報を取得するために使用できます。 この verb は、現在実行中のすべての TP に関する情報を戻します。この情報には、現在 APPC verb または CPI-C 呼び出しがあるかどうかにかかわらず、新規の着信会話を受け入れるための未処理の呼び出しが

この verb は実行中のノードに対して発行する必要があります

VCB 構造体

```
typedef struct query_available_tp
{
    AP_UINT16         opcode;             /* Verb operation code           */
    unsigned char     reserv2;            /* reserved                      */
    unsigned char     format;             /* reserved                      */
    AP_UINT16         primary_rc;         /* Primary return code           */
    AP_UINT32         secondary_rc;       /* Secondary return code         */
    unsigned char     *buf_ptr;           /* pointer to buffer             */
    AP_UINT32         buf_size;           /* buffer size                   */
    AP_UINT32         total_buf_size;     /* total buffer size required    */
    AP_UINT16         num_entries;        /* number of entries             */
    AP_UINT16         total_num_entries;  /* total number of entries       */
    unsigned char     list_options;       /* listing options               */
    unsigned char     reserv3[3];         /* reserved                      */
    unsigned char     tp_name[64];        /* TP name                       */
    unsigned char     system_name[128];   /* computer name where TP is     */
                                          /*  running                      */
} QUERY_AVAILABLE_TP;
```

```
typedef struct available_tp_data
{
    AP_UINT16         overlay_size;       /* size of returned entry        */
    unsigned char     tp_name[64];        /* TP name                       */
    unsigned char     reserv4[4];         /* reserved                      */
    unsigned char     system_name[128];   /* computer name where TP is     */
                                          /*  running                      */
} AVAILABLE_TP_DATA;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
使用可能な照会の追加 （ _T ）

buf_ptr
CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

num_entries
データが戻される TP の最大数。 範囲ではなく、特定の TP のデータを要求するには、 1 という値を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、 CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータの戻りを開始する TP のリスト内の位置。 可能な値は次のとおりです

リストの最初のリスト （ _R ）
リストの最初の項目から開始します。

リストを含む （ 包括的 ）
TP 名とシステム名の組み合わせによって指定されたエントリーから開始します。

次への AP_LIST_FROM_NEXT
TP 名とシステム名の組み合わせによって指定されたエントリーの直後のエントリーから開始します。

リストの順序とアプリケーションでの特定のエントリーの取得方法について詳しくは、 34 ページの『List options for QUERY_* Verbs』を参照してください。

Tp_name
TP 名。 これは 64 バイトのストリングで、名前が 64 文字より短い場合は、右側にスペースが埋め込まれます。 *list_options* を **リストの最初のリスト** （ _R ）に設定すると、この値は無視されます。

システム名
TP 情報が必要なコンピューター名。 システム名は、 1 から 128 文字の ASCII ストリングです。これは CS Linux コンピューター名と一致しなければなりません。 *list_options* を **リストの最初のリスト** （ _R ）に設定すると、この値は無視されます。

コンピューター名に . (ピリオド) 文字が含まれている場合、 CS Linux は、それが完全修飾名であると見なします。それ以外の場合は、 DNS ルックアップを実行してコンピューター名を判別します。

CS Linux が単一コンピューター上のすべてのプログラムで実行されている場合は、コンピューター名を指定する必要はありません (すべて 2 進ゼロのままにすることができます)。 クライアント/サーバー・システムの場合は、指定されたコンピューター上の TP のみをリストするコンピューター名を指定するか、すべてのコンピューター上の TP をリストするためのすべての 2 進ゼロとしてそのままにします。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

available_tp_data.overlay_size
The size of the returned `available_tp_data` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `available_tp_data` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

available_tp_data.tp_name
TP name. This is a 64-byte string, padded on the right with spaces if the name is shorter than 64 characters.

available_tp_data.system_name
Name of the computer where the TP is running.

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

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

AP_UNKNOWN_TP
The *list_options* parameter was set to AP_LIST_INCLUSIVE, to list all entries starting from the supplied name, but the *tp_name* parameter was not valid.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

照会バッファーの可用性

この verb は、 CS Linux が現在使用している STREAMS バッファーの量、使用されている最大量、および使用可能な最大量 (SET_BUFFER_AVAILABILITY verb を使用して指定される) の量に関する情報を戻します。 これにより、 STREAMS バッファーの使用状況を確認し、制限を適切に設定して、 CS Linux コンポーネント、および Linux コンピューター上の他のプログラム用に十分なバッファーが使用可能になるようにすることができます。 また、この verb は、 CS Linux サポート担当者が使用するために、バッファー使用に関連する追加の内部値を戻します。

VCB 構造体

```
typedef struct query_buffer_availability
{
   AP_UINT16        opcode;                /* verb operation code           */
   unsigned char    reserv2;               /* reserved                      */
   unsigned char    format;                /* reserved                      */
   AP_UINT16        primary_rc;            /* primary return code           */
   AP_UINT32        secondary_rc;          /* secondary return code         */
   AP_UINT32        reset_max_values;      /* set stored max values to      */
                                           /*  current                      */
   AP_UINT32        buf_avail;             /* maximum buffer space          */
                                           /*  available                    */
   AP_UINT32        buf_total_count;       /* current buffer usage - count  */
   AP_UINT32        buf_total_bytes;       /* current buffer usage - bytes  */
   AP_UINT32        buf_rsrv_count;        /* buffers reserved - count      */
   AP_UINT32        buf_rsrv_bytes[2];     /* buffers reserved - bytes      */
   AP_UINT32        buf_res_use_count;     /* usage of reserved buffers     */
                                           /*  - count                      */
   AP_UINT32        buf_res_use_bytes;     /* usage of reserved buffers     */
                                           /*  - bytes                      */
   AP_UINT32        peak_usage;            /* peak usage                    */
   AP_UINT32        peak_decay;            /* peak decay                    */
   unsigned char    throttle_status;       /* throttle status               */
   unsigned char    buf_use_status;        /* congestion status             */
   AP_UINT32        max_buf_total_count;   /* maximum buffer usage - count  */
   AP_UINT32        max_buf_total_bytes;   /* maximum buffer usage - bytes  */
```

```
    AP_UINT32        max_buf_rsrv_count;     /* max buffers reserved - count  */
    AP_UINT32        max_buf_rsrv_bytes[2];  /* max buffers reserved - bytes  */
    AP_UINT32        max_buf_res_use_count;  /* max rsrv buffer usage - count */
    AP_UINT32        max_buf_res_use_bytes;  /* max rsrv buffer usage - bytes */
    AP_UINT32        max_peak_usage;         /* maximum peak usage            */
    unsigned char    max_throttle_status;    /* maximum throttle status       */
    unsigned char    max_buf_use_status;     /* maximum congestion status     */
    unsigned char    debug_param[32];        /* reserved                      */
    unsigned char    reserv3[8];             /* reserved                      */
} QUERY_BUFFER_AVAILABILITY;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_BUFFER_AVAILABILITY

reset_max_values
Specify whether CS Linux should reset the values for the *max_** parameters (after returning them on this verb) to match the current values of these parameters. This ensures that a subsequent QUERY_BUFFER_AVAILABILITY verb will return the maximum values reached since this verb, rather than the maximum values reached since the system was started (or since the values for the *max_** parameters were last reset). Possible values are:

AP_YES
Reset the values for the *max_** parameters to match the current values.

AP_NO
Do not reset the values for the *max_** parameters.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters. Values returned in other fields are for use by CS Linux support personnel.

primary_rc
AP_OK

secondary_rc
Not used.

buf_avail
The maximum amount of STREAMS buffer space available to CS Linux, in bytes, as defined by a SET_BUFFER_AVAILABILITY verb.

buf_total_count
The number of buffers currently allocated to CS Linux components.

buf_total_bytes
The total amount of storage in buffers currently allocated to CS 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:

- AP_CONGESTED
- AP_UNCONGESTED

max_buf_total_count
The maximum number of buffers that have been allocated to CS Linux components at any time.

max_buf_total_bytes
The maximum amount of buffer storage that has been allocated to CS Linux components at any 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:

- AP_CONGESTED
- AP_UNCONGESTED

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

照会集中ログ・ロガー

この verb は、中央ロガーとして現在定義されているノードの名前を戻します (すべてのサーバーからの CS Linux ログ・メッセージが送信される中央ログ・ファイルを保持しているノード)。 この verb は、中央ロギングがアクティブかどうかに関する情報を戻しません。 QUERY_CENTRAL_LOGGING を使用してこれを判別してください。

この verb は、ヌルのターゲット・ハンドルで発行する必要があります

VCB 構造体

```
typedef struct query_central_logger
{
     AP_UINT16          opcode;             /* verb operation code          */
     unsigned char      reserv2;            /* reserved                     */
     unsigned char      format;             /* reserved                     */
     AP_UINT16          primary_rc;         /* primary return code          */
     AP_UINT32          secondary_rc;       /* secondary return code        */
     unsigned char      reserv3[4];         /* reserved                     */
     unsigned char      node_name[128];     /* name of central logger       */
} QUERY_CENTRAL_LOGGER;
```

Supplied parameters

The application supplies the following parameter:

opcode
AP_QUERY_CENTRAL_LOGGER

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

secondary_rc
未使用。

ノード名
中央ロガーとして定義されているノードの名前。

Returned parameters: state check

If the verb does not execute because of a state error, CS Linux returns the following parameters:

primary_rc
AP_STATE_CHECK

secondary_rc

AP_NO_CENTRAL_LOG
No controller server is currently active.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

照会集中ロギング

この verb は、 CS Linux ログ・メッセージがすべてのサーバーから中央ファイルに送信されるか、または各サーバー上の個別のファイルに送信されるかに関する情報を戻します。 詳しくは、 573 ページの『SET_LOG_FILE』を参照してください。

この verb は、現在中央ロガーとして動作しているノードに対して発行される必要があります。このノードへのアクセスについては、 53 ページの『接続ノード』を参照してください。

VCB structure

```
typedef struct query_central_logging
{
     AP_UINT16          opcode;          /* verb operation code            */
     unsigned char      reserv2;         /* reserved                       */
     unsigned char      format;          /* reserved                       */
     AP_UINT16          primary_rc;      /* primary return code            */
     AP_UINT32          secondary_rc;    /* secondary return code          */
     unsigned char      enabled;         /* is central logging enabled?    */
     unsigned char      reserv3[3];      /* reserved                       */
} QUERY_CENTRAL_LOGGING;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
AP_QUERY_CENTRAL_LOGGING

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

secondary_rc
未使用。

有効
中央ロギングを使用可能または使用不可にするかどうか 可能な値は次のとおりです

類人猿
セントラル・ロギングは使用可能です。 すべてのログ・メッセージは、中央ロガーとして現在活動しているノード上の単一ファイルに送信されます。

アプ・ノー
中央ロギングは使用不可です。 各サーバーからのログ・メッセージは、そのサーバー上のファイル (SET_LOG_FILE verb を使用して指定) に送信されます。

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc

AP_NOT_CENTRAL_ロガー
この verb は、中央ロガーではないノードに対して発行されました。

状態チェック

状態エラーが原因でコマンドが実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc

集中ログの追加
現在アクティブなコントローラー・サーバーはありません。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

照会 (CN)

QUERY_CN は、隣接接続ネットワークに関する情報を戻します。 この情報は、 "決定データ" (実行中に動的に収集されたデータ) と "定義済みデータ" (DEFINE_CN 上のアプリケーションによって提供されるデータ) として構造化されます。

この verb は、使用されるオプションに応じて、特定の接続ネットワークまたは複数の接続ネットワークに関する情報を取得するために使用できます。 このメッセージは、ネットワーク・ノードまたはエンド・ノードでのみ発行できます。 LEN ノードでは無効です。

VCB structure

```
typedef struct query_cn
{
   AP_UINT16         opcode;             /* Verb operation code            */
   unsigned char     reserv2;            /* reserved                       */
```

```
   unsigned char     format;              /* reserved                        */
   AP_UINT16         primary_rc;          /* Primary return code             */
   AP_UINT32         secondary_rc;        /* Secondary return code           */
   unsigned char     *buf_ptr;            /* pointer to buffer               */
   AP_UINT32         buf_size;            /* buffer size                     */
   AP_UINT32         total_buf_size;      /* total buffer size required      */
   AP_UINT16         num_entries;         /* number of entries               */
   AP_UINT16         total_num_entries;   /* total number of entries         */
   unsigned char     list_options;        /* listing options                 */
   unsigned char     reserv3;             /* reserved                        */
   unsigned char     fqcn_name[17];       /* Name of Connection Network      */
} QUERY_CN;
```

```
typedef struct cn_data
{
   AP_UINT16         overlay_size;        /* size of returned entry          */
   unsigned char     fqcn_name[17];       /* Name of Connection Network      */
   unsigned char     reserv1;             /* reserved                        */
   CN_DET_DATA       det_data;            /* Determined data                 */
   CN_DEF_DATA       def_data;            /* Defined data                    */
} CN_DATA;
```

```
typedef struct cn_det_data
{
   AP_UINT16         num_act_ports;       /* number of active ports          */
   unsigned char     reserva[20];         /* reserved                        */
} CN_DET_DATA;
```

```
typedef struct cn_def_data
{
   unsigned char     description[32];     /* resource description            */
   unsigned char     reserve0[16];        /* reserved                        */
   unsigned char     num_ports;           /* number of ports on CN           */
   unsigned char     cn_type;             /* reserved                        */
   unsigned char     ipv6_addr_only;      /* use IPv6 address                */
   unsigned char     reserve1[14];        /* reserved                        */
   TG_DEFINED_CHARS  tg_chars;            /* TG characteristics              */
} CN_DEF_DATA;
```

```
typedef struct tg_defined_chars
{
   unsigned char     effect_cap;          /* effective capacity              */
   unsigned char     reserve1[5];         /* reserved                        */
   unsigned char     connect_cost;        /* connection cost                 */
   unsigned char     byte_cost;           /* byte cost                       */
   unsigned char     reserve2;            /* reserved                        */
   unsigned char     security;            /* security                        */
   unsigned char     prop_delay;          /* propagation delay               */
   unsigned char     modem_class;         /* reserved                        */
   unsigned char     user_def_parm_1;     /* user-defined parameter 1        */
   unsigned char     user_def_parm_2;     /* user-defined parameter 2        */
   unsigned char     user_def_parm_3;     /* user-defined parameter 3        */
} TG_DEFINED_CHARS;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
アプ・クエリーの CN

buf_ptr
CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

num_entries
データが戻される CNS の最大数。 範囲ではなく、特定の CN のデータを要求するには、 1 という値を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、 CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータの戻りを開始する必要のある、 CNs のリスト内の位置。 可能な値は次のとおりです

リストの最初のリスト （ _R ）
リストの最初の項目から開始します。

リストを含む （ 包括的 ）
fqcn_name パラメーターで指定されたエントリーから開始します。

次への AP_LIST_FROM_NEXT
fqcn_name パラメーターによって指定されたエントリーの直後のエントリーから開始します。

リストの順序とアプリケーションでの特定のエントリーの取得方法について詳しくは、 34 ページの『List options for QUERY_* Verbs』を参照してください。

fqcn_name
情報が必要な CN の完全修飾名、または CNs のリストの索引として使用される名前。 *list_options* を **リストの最初のリスト （ _R ）**に設定すると、この値は無視されます。

この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

cn_data.overlay_size
The size of the returned `cn_data` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `cn_data` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

cn_data.fqcn_name
Fully qualified name of the CN. The name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of 1-8 A-string characters, an EBCDIC dot (period) character, and a network name of 1-8 A-string characters.

cn_data.det_data.num_act_ports
The number of active ports on the connection network.

cn_data.def_data.description
A null-terminated text string describing the CN, as specified in the definition of the CN.

cn_data.def_data.num_ports
The total number of ports on the connection network.

cn_data.def_data.ipv6_addr_only
For a Connection Network on an IPv6 network for HPR/IP, this parameter indicates if the IP addressing for the Connection Network uses IPv6 DNS names only or IPv6 addresses only. Possible values are:

YES
IP addressing for the Connection Network uses IPv6 addresses only.

NO
IP addressing for the Connection Network uses IPv6 DNS names only.

cn_data.def_data.tg_chars.effect_cap
Actual bits per second rate (line speed). The value is encoded as a 1-byte floating point number, represented by the formula 0.1 mmm * 2 ^{eeeee} where the bit representation of the byte is eeeeemmm. Each unit of effective capacity is equal to 300 bits per second.

cn_data.def_data.tg_chars.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.

cn_data.def_data.tg_chars.byte_cost
Cost per byte. Values are integers in the range 0-255, where zero is the lowest cost per byte and 255 is the highest.

cn_data.def_data.tg_chars.security
Security level of the network. Possible values are:

AP_SEC_NONSECURE
No security.

AP_SEC_PUBLIC_SWITCHED_NETWORK
Data is transmitted over a public switched network.

AP_SEC_UNDERGROUND_CABLE
Data is transmitted over secure underground cable.

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

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

AP_SEC_ENCRYPTED
Data is encrypted before transmission over the line.

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

AP_SEC_MAXIMUM
Maximum security.

cn_data.def_data.tg_chars.prop_delay
Propagation delay: the time that a signal takes to travel the length of the link. Possible values are:

AP_PROP_DELAY_MINIMUM
Minimum propagation delay.

AP_PROP_DELAY_LAN
Delay is less than 480 microseconds (typical for a LAN).

AP_PROP_DELAY_TELEPHONE
Delay is in the range 480-49,512 microseconds (typical for a telephone network).

AP_PROP_DELAY_PKT_SWITCHED_NET
Delay is in the range 49,512-245,760 microseconds (typical for a packet-switched network).

AP_PROP_DELAY_SATELLITE
Delay is greater than 245,760 microseconds (typical for a satellite link).

AP_PROP_DELAY_MAXIMUM
Maximum propagation delay.

cn_data.def_data.tg_chars.user_def_parm_1* through *def_data.tg_chars.user_def_parm_3
User-defined parameters, which include other TG characteristics not covered by the above parameters. Each of these parameters is set to a value in the range 0-255.

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

ファイル・ファイル名の変更
list_options パラメーターが **リストを含む** （ **包括的** ）に設定されました。指定された名前から始まるすべての項目をリストしますが、 *fqcn_name* パラメーターが無効でした。

ファイルの追加リスト・オプション
list_options パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: function not supported

If the verb does not execute successfully because the local node is a LEN node, CS Linux returns the following parameters:

primary_rc

AP_FUNCTION_NOT_SUPPORTED
The local node is a LEN node. This verb is valid only at a network node or an end node.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

照会 _CN_PORT

QUERY_CN_PORT は、隣接接続ネットワーク上で定義されたポートに関する情報を戻します。

この verb は、使用されるオプションに応じて、特定のポートに関する情報を取得したり、複数のポートに関する情報を取得したりするために使用できます。 このメッセージは、ネットワーク・ノードまたはエンド・ノードでのみ発行できます。 LEN ノードでは無効です。

VCB 構造体

```
typedef struct query_cn_port
{
     AP_UINT16          opcode;             /* Verb operation code          */
     unsigned char      reserv2;            /* reserved                     */
     unsigned char      format;             /* reserved                     */
     AP_UINT16          primary_rc;         /* Primary return code          */
     AP_UINT32          secondary_rc;       /* Secondary return code        */
     unsigned char     *buf_ptr;            /* pointer to buffer            */
     AP_UINT32          buf_size;           /* buffer size                  */
     AP_UINT32          total_buf_size;     /* total buffer size required   */
     AP_UINT16          num_entries;        /* number of entries            */
     AP_UINT16          total_num_entries;  /* total number of entries      */
     unsigned char      list_options;       /* listing options              */
     unsigned char      reserv3;            /* reserved                     */
     unsigned char      fqcn_name[17];      /* Name of Connection Network   */
```

```
     unsigned char      port_name[8];        /* port name                    */
} QUERY_CN_PORT;
```

```
typedef struct cn_port_data
{
     AP_UINT16          overlay_size;        /* size of returned entry       */
     unsigned char      fqcn_name[17];       /* Name of Connection Network   */
     unsigned char      port_name[8];        /* name of port                 */
     unsigned char      tg_num;              /* transmission group number    */
     unsigned char      reserva[20];         /* reserved                     */
} CN_PORT_DATA;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
照会の CN_PORT

buf_ptr
CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

num_entries
データが戻されるポートの最大数。 特定の範囲ではなく、特定のポートのデータを要求するには、 1 という値を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、 CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータの戻りを開始する必要のあるポートのリスト内の位置。 可能な値は次のとおりです

リストの最初のリスト （ _R ）
リストの最初の項目から開始します。

リストを含む （ 包括的 ）
ポート名 パラメーターで指定されたエントリーから開始します。

次への AP_LIST_FROM_NEXT
ポート名 パラメーターによって指定されたエントリーの直後のエントリーから開始します。

リストの順序とアプリケーションでの特定のエントリーの取得方法について詳しくは、 34 ページの『List options for QUERY_* Verbs』を参照してください。

fqcn_name
必要なポートが定義されている CN の完全修飾名、またはポートのリストを必要とする CN の完全修飾名。

この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

ポート名
情報が必要なポートの名前、またはポートのリストへの索引として使用される名前。 これは 8 バイトの ASCII ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

buf_size
提供されたバッファーに戻された情報の長さ。

total_buf_size
要求されたすべてのリスト情報を戻すために必要となるバッファーのサイズを示す戻り値。 *buf_size* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

num_entries
データ・バッファーに戻されたエントリーの数。

total_num_entries
使用可能な項目の合計数。 *num_entries* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

データ・バッファー内の各項目は、以下のパラメーターで構成されます。

cn_port_data.オーバーレイ ・サイズ
戻された cn_port_data 構造体のサイズ。すなわち、データ・バッファー内の次のエントリーの先頭までのオフセット。

アプリケーションが戻されたバッファーを調べて、各 cn_port_data 構造体を順番に検出する必要がある場合は、この値を使用して次のデータ構造の正しいオフセットに移動しなければなりません。また、 C sizeof() 演算子を使用してはなりません。 これは、返されるオーバーレイのサイズが、 CS Linux の将来のリリースで増加する可能性があるためです。戻されたオーバーレイ ・サイズを使用すると、アプリケーションが今後のリリースで引き続き動作することが保証されます。

cn_port_data.fqcn_name
CN の完全修飾名。 この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、 1 から 8 文字の A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および 1 から 8 文字の A ストリング文字のネットワーク名で構成されます。

cn_port_data.port_name
ポートの名前。 これは 8 バイトの ASCII ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

cn_port_data.tg_num
指定されたポートの伝送グループ番号。

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

ファイル・ ファイル名の変更
list_options パラメーターが **リストを含む** (**包括的**)に設定されました。指定された名前から始まるすべての項目をリストしますが、 *fqcn_name* パラメーターが無効でした。

ポートフォリオ・ ポート名
list_options パラメーターが **リストを含む** (**包括的**)に設定されました。指定された名前から始まるすべての項目をリストしますが、 ポート名 パラメーターが無効でした。

ファイルの追加リスト・ オプション
list_options パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: function not supported

If the verb does not execute successfully because the local node is a LEN node, CS Linux returns the following parameters:

primary_rc

AP_FUNCTION_NOT_SUPPORTED
The local node is a LEN node. This verb is valid only at a network node or an end node.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

照会会話

QUERY_CONVERSATION は、特定のローカル LU を使用した会話に関する情報を戻します。

この verb は、使用されるオプションに応じて、特定の会話または会話の範囲に関する情報を取得するために使用できます。

この verb は実行中のノードに対して発行する必要があります

VCB 構造体

```
typedef struct query_conversation
{
  AP_UINT16      opcode;                  /* verb operation code        */
  unsigned char  reserv2;                 /* reserved                   */
  unsigned char  format;                  /* reserved                   */
  AP_UINT16      primary_rc;              /* primary return code        */
  AP_UINT32      secondary_rc;            /* secondary return code      */
  unsigned char *buf_ptr;                 /* pointer to buffer          */
  AP_UINT32      buf_size;                /* buffer size                */
  AP_UINT32      total_buf_size;          /* total buffer size required */
  AP_UINT16      num_entries;             /* number of entries          */
  AP_UINT16      total_num_entries;       /* total number of entries    */
  unsigned char  list_options;            /* listing options            */
  unsigned char  reserv3;                 /* reserved                   */
  unsigned char  lu_name[8];              /* LU Name                    */
  unsigned char  lu_alias[8];             /* LU Alias                   */
  AP_UINT32      conv_id;                 /* Conversation ID            */
  unsigned char  session_id[8];           /* Session ID                 */
  unsigned char  reserv4[12];             /* reserved                   */
} QUERY_CONVERSATION;
```

```
typedef struct conv_summary
{
  AP_UINT16      overlay_size;            /* overlay size               */
  AP_UINT32      conv_id;                 /* conversation ID            */
  unsigned char  local_tp_name[64];       /* local TP name              */
  unsigned char  partner_tp_name[64];     /* partner TP name            */
  unsigned char  tp_id[8];                /* TP ID                      */
  unsigned char  sess_id[8];              /* Session ID                 */
  AP_UINT32      conv_start_time;         /* Conversation start time    */
  AP_UINT32      bytes_sent;              /* Number of bytes sent       */
  AP_UINT32      bytes_received;          /* Number of bytes received   */
  unsigned char  conv_state;              /* conversation state         */
  unsigned char  duplex_type;             /* full- or half-duplex conv? */
} CONV_SUMMARY;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_CONVERSATION

buf_ptr
A pointer to a data buffer that CS Linux will use to return the requested information.

buf_size
Size of the supplied data buffer.

num_entries
Maximum number of conversations for which data should be returned. To request data for a specific conversation rather than a range, specify the value 1. To return as many entries as possible, specify zero; in this case, CS Linux will return the maximum number of entries that can be accommodated in the supplied data buffer.

list_options
The position in the list from which CS Linux should begin to return data. Specify one of the following values:

AP_FIRST_IN_LIST
Start at the first entry in the list.

AP_LIST_INCLUSIVE
Start at the entry specified by the combination of local LU and conversation ID.

AP_LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the combination of local LU and conversation ID.

The combination of the local LU (*lu_name* or *lu_alias*) and conversation ID (*conv_id*) specified is used as an index into the list of sessions if the *list_options* parameter is set to AP_LIST_INCLUSIVE or AP_LIST_FROM_NEXT.

For more information about how the list is ordered and how the application can obtain specific entries from it, see "List options for QUERY_* Verbs" on page 34.

lu_name
LU name. This is an 8-byte type-A EBCDIC string, padded on the right with spaces if the name is shorter than 8 characters. To specify that the LU is identified by its alias rather than its LU name, set this parameter to 8 binary zeros and specify the LU alias in the following parameter. To specify the LU associated with the local CP (the default LU), set both *lu_name* and *lu_alias* to binary zeros.

lu_alias
Locally defined LU alias. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes. This parameter is used only if *lu_name* is set to 8 binary zeros; it is ignored otherwise. To specify the LU associated with the local CP (the default LU), set both *lu_name* and *lu_alias* to binary zeros.

conv_id
Identifier of the conversation for which information is required, or the conversation ID to be used as an index into the list of conversations. The conversation ID was returned by the ALLOCATE verb in the invoking TP, or by the RECEIVE_ALLOCATE verb in the invoked TP.

This parameter is ignored if *list_options* is set to AP_FIRST_IN_LIST.

session_id
8-byte identifier of the session. To list only information about conversations associated with a specific session, specify the session identifier. To obtain a complete list for all sessions, set this field to binary zeros.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

conv_summary.overlay_size
The size of the returned `conv_summary` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `conv_summary` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

conv_summary.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.

conv_summary.local_tp_name
The name of the local TP in the conversation.

conv_summary.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.

conv_summary.tp_id
The TP identifier of the conversation.

conv_summary.session_id
The session identifier of the session allocated to the conversation.

conv_summary.conv_start_time
The elapsed time in hundredths of seconds between the time when the CS Linux node was started and the time when the conversation was started.

conv_summary.bytes_sent
The number of bytes that have been sent from the local TP to the partner TP since the start of the conversation.

conv_summary.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_summary.conv_state
The current state of the conversation. Values for a half-duplex conversation:

- AP_CONFIRM_STATE
- AP_CONFIRM_DEALL_STATE
- AP_CONFIRM_SEND_STATE
- AP_END_CONV_STATE
- AP_PEND_DEALL_STATE
- AP_PEND_POST_STATE
- AP_POST_ON_RECEIPT_STATE
- AP_RECEIVE_STATE
- AP_RESET_STATE
- AP_SEND_STATE
- AP_SEND_PENDING_STATE

Values for a full-duplex conversation:

- AP_RESET_STATE
- AP_SEND_ONLY_STATE
- AP_SEND_RECEIVE_STATE
- AP_RECEIVE_ONLY_STATE

conv_summary.duplex_type
The duplex type of the conversation. Values:

- AP_HALF_DUPLEX
- AP_FULL_DUPLEX

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_BAD_CONV_ID
The *list_options* parameter was set to AP_LIST_INCLUSIVE to list all entries starting from the supplied value, but the *conv_id* parameter was not valid.

AP_INVALID_LU_ALIAS
The specified *lu_alias* parameter was not valid.

AP_INVALID_LU_NAME
The specified *lu_name* parameter was not valid.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

クエリー・コア

QUERY_COS は、特定のサービス・クラス (COS) の経路計算情報を戻します。

この verb は、使用されるオプションに応じて、特定の COS または複数の COS に関する情報を取得するために使用できます。

VCB 構造体

```
typedef struct query_cos
{
   AP_UINT16        opcode;               /* verb operation code          */
   unsigned char    reserv2;              /* reserved                     */
   unsigned char    format;               /* reserved                     */
   AP_UINT16        primary_rc;           /* primary return code          */
   AP_UINT32        secondary_rc;         /* secondary return code        */
   unsigned char    *buf_ptr;             /* pointer to buffer            */
   AP_UINT32        buf_size;             /* buffer size                  */
   AP_UINT32        total_buf_size;       /* total buffer size required   */
   AP_UINT16        num_entries;          /* number of entries            */
   AP_UINT16        total_num_entries;    /* total number of entries      */
   unsigned char    list_options;         /* listing options              */
```

```
   unsigned char    reserv3;               /* reserved                     */
   unsigned char    cos_name[8];           /* cos name                     */
} QUERY_COS;
```

```
typedef struct cos_data
{
   AP_UINT16        overlay_size;          /* size of returned entry       */
   unsigned char    cos_name[8];           /* cos name                     */
   unsigned char    description[32];       /* resource description         */
   unsigned char    reserv1[16];           /* reserved                     */
   unsigned char    transmission_priority; /* transmission priority        */
   AP_UINT16        num_of_node_rows;      /* number of node rows          */
   AP_UINT16        num_of_tg_rows;        /* number of tg rows            */
   AP_UINT32        trees;                 /* number of tree caches for COS */
   AP_UINT32        calcs;                 /* number of route calculations  */
                                           /*  for this COS                */
   AP_UINT32        rejs;                  /* number of route rejects for   */
                                           /*  COS                         */
   unsigned char    reserva[20];           /* reserved                     */
} COS_DATA;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_COS

buf_ptr
A pointer to a data buffer that CS Linux will use to return the requested information.

buf_size
Size of the supplied data buffer.

num_entries
Maximum number of COSs for which data should be returned. To request data for a specific COS rather than a range, specify the value 1. To return as many entries as possible, specify zero; in this case, CS Linux will return the maximum number of entries that can be accommodated in the supplied data buffer.

list_options
The position in the list of COSs from which CS Linux should begin to return data. Possible values are:

AP_FIRST_IN_LIST
Start at the first entry in the list.

AP_LIST_INCLUSIVE
Start at the entry specified by the *cos_name* parameter.

AP_LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *cos_name* parameter.

For more information about how the list is ordered and how the application can obtain specific entries from it, see “List options for QUERY_* Verbs” on page 34.

cos_name
Class of service 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 AP_FIRST_IN_LIST. The name is an 8-byte type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces if the name is shorter than 8 characters.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

buf_size
提供されたバッファーに戻された情報の長さ。

total_buf_size
要求されたすべてのリスト情報を戻すために必要となるバッファーのサイズを示す戻り値。 *buf_size* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

num_entries
データ・バッファーに戻されたエントリーの数。

total_num_entries
使用可能な項目の合計数。 *num_entries* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

データ・バッファー内の各項目は、以下のパラメーターで構成されます。

cos_data.オーバーレイ ・サイズの
戻された cos_data **データ** 構造体のサイズ。すなわち、データ・バッファー内の次のエントリーの先頭までのオフセット。

アプリケーションが戻されたバッファーを調べて、各 cos_data **データ** 構造体を順番に検出する必要がある場合は、この値を使用して次のデータ構造の正しいオフセットに移動しなければなりません。また、 C sizeof() 演算子を使用してはなりません。 これは、返されるオーバーレイのサイズが、 CS Linux の将来のリリースで増加する可能性があるためです。戻されたオーバーレイ ・サイズを使用すると、アプリケーションが今後のリリースで引き続き動作することが保証されます。

cos_data.cos_name
サービス・クラス名。 これは、 8 バイトの英数字のタイプ A の EBCDIC ストリング (文字で始まる) で、右側に EBCDIC のスペースが埋め込まれます。

cos_data.description
COS の定義に指定されている、 COS を記述するヌル終了のテキスト・ストリング。

cos_data.transmission_priority
伝送優先順位。 可能な値は次のとおりです

低価格

メディア・ メディア

上位 （ 上位 ）

ネットワーク (最高の優先順位)

ノード内の余弦の行数
この COS に定義されたノード行の数。

***tg_tg_tg_rows* の余力**
この COS に定義された TG 行の数。

cos_data.tree
最後の初期化以降にこの COS 用に作成された経路ツリー・キャッシュの数。

cos_data.calcs
このクラスのサービスを指定するセッション活動化要求の数 (つまり、経路計算) 。

cos_data.rejs
ネットワークを介してこのノードから指定された宛先までの受け入れ可能な経路がなかったために失敗したセッション活動化要求の数。 経路は、指定されたサービス・クラスを提供することができるアクティブな TG およびノードで完全に構成されている場合にのみ受け入れられます。

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_COS_NAME
The *list_options* parameter was set to AP_LIST_INCLUSIVE, to list all entries starting from the supplied name, but the *cos_name* parameter was not valid.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

QUERY_COS_NODE_ROW

QUERY_COS_NODE_ROW returns node row information for a specified class of service as previously defined by DEFINE_COS (or implicitly by the node for the SNA-defined COSs).

This verb can be used to obtain information about a specific COS node row, or about multiple COS node rows, depending on the options used.

VCB 構造体

```
typedef struct query_cos_node_row
{
    AP_UINT16          opcode;              /* verb operation code          */
    unsigned char      reserv2;             /* reserved                     */
    unsigned char      format;              /* reserved                     */
    AP_UINT16          primary_rc;          /* primary return code          */
    AP_UINT32          secondary_rc;        /* secondary return code        */
    unsigned char      *buf_ptr;            /* pointer to buffer            */
    AP_UINT32          buf_size;            /* buffer size                  */
    AP_UINT32          total_buf_size;      /* total buffer size required   */
    AP_UINT16          num_entries;         /* number of entries            */
    AP_UINT16          total_num_entries;   /* total number of entries      */
    unsigned char      list_options;        /* listing options              */
    unsigned char      reserv3;             /* reserved                     */
    unsigned char      cos_name[8];         /* cos name                     */
    AP_UINT16          node_row_index;      /* node row index               */
} QUERY_COS_NODE_ROW;
```

```
typedef struct cos_node_row_data
{
    AP_UINT16          overlay_size;        /* size of returned entry       */
    unsigned char      cos_name[8];         /* cos name                     */
    AP_UINT16          node_row_index;      /* node row index               */
    COS_NODE_ROW       node_row;            /* cos node row information     */
} COS_NODE_ROW_DATA;
```

```
typedef struct cos_node_row
{
    COS_NODE_STATUS    minimum;             /* minimum                      */
    COS_NODE_STATUS    maximum;             /* maximum                      */
    unsigned char      weight;              /* weight                       */
    unsigned char      reserv1;             /* reserved                     */
} COS_NODE_ROW;
```

```
typedef struct cos_node_status
{
    unsigned char      rar;                 /* route additional resistance  */
    unsigned char      status;              /* node status                  */
    unsigned char      reserv1[2];          /* reserved                     */
} COS_NODE_STATUS;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
追加ノード· ノード行の追加

buf_ptr
CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

num_entries
データが戻される COS ノード行の最大数。 範囲ではなく、特定の COS ノード行のデータを要求するには、 1 という値を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、 CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータの戻りを開始する、 COS ノード行のリスト内の位置。 可能な値は次のとおりです

リストの最初のリスト （ _R ）
リストの最初の項目から開始します。

リストを含む （ 包括的 ）
cos_name パラメーターと ノード行索引 パラメーターの組み合わせによって指定されたエントリーから開始します。

次への AP_LIST_FROM_NEXT
cos_name パラメーターと ノード行索引 パラメーターの組み合わせによって指定されたエントリーの直後のエントリーから開始します。

このリストは、 *cos_name* によって配列され、その後 COS ごとに ノード行索引 によって配列されます。 アプリケーションがリストから特定の項目を取得する方法について詳しくは、 34 ページの『List options for QUERY_* Verbs』を参照してください。

cos_name
ノード行情報が必要なサービス・クラス名、またはリスト内の索引として使用する名前。 *list_options* を **リストの最初のリスト** （ _R ）に設定すると、この値は無視されます。 この名前は、 8 バイトのタイプ A の EBCDIC ストリング (文字で始まる) で、名前が 8 文字より短い場合は、右側に EBCDIC のスペースが埋め込まれます。

ノード行索引
情報が必要なノード行番号、またはリスト内の索引として使用される番号。 *list_options* を **リストの最初のリスト** （ _R ）に設定すると、この値は無視されます。 この COS に関連付けられているノード行の数を判別するには、 QUERY_COS を使用します。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

buf_size
提供されたバッファーに戻された情報の長さ。

total_buf_size
要求されたすべてのリスト情報を戻すために必要となるバッファーのサイズを示す戻り値。 *buf_size* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

num_entries
データ・バッファーに戻されたエントリーの数。

total_num_entries
使用可能な項目の合計数。 *num_entries* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

データ・バッファー内の各項目は、以下のパラメーターで構成されます。

***cos_node_row_data.*オーバーレイ・サイズ**
戻された cos_node_row_data の構造体のサイズ。すなわち、データ・バッファー内の次のエントリーの先頭までのオフセット。

アプリケーションが戻されたバッファーを調べて、各 cos_node_row_data の構造体を順番に検出する必要がある場合は、この値を使用して次のデータ構造の正しいオフセットに移動しなければなりません。また、C sizeof() 演算子を使用してはなりません。これは、返されるオーバーレイのサイズが、CS Linux の将来のリリースで増加する可能性があるためです。戻されたオーバーレイ・サイズを使用すると、アプリケーションが今後のリリースで引き続き動作することが保証されます。

cos_node_row_data.cos_name
サービス・クラス名。これは 8 バイトのタイプ A の EBCDIC ストリング (文字で始まる) で、名前が 8 文字より短い場合は、右側に EBCDIC のスペースが埋め込まれます。

cos_node_row_data.node_row_index
ノード行索引。

cos_node_row_data.node_row.minimum.rar
追加の抵抗最小値を、0-255 の範囲で経路指定します。

cos_node_row_data.node_row.minimum.status
ノードの最小輻輳状況を指定します。このパラメーターは、**輻輳しています**されているその他の値のいずれかに、または論理 **それとも**を使用して結合された 2 つ以上のその他の値に設定することができます。可能な値は次のとおりです

輻輳しています
ISR セッションの数が、ノードの構成の *isr_sessions_upper_threshold* 値よりも低くなっています。

輻輳 (しげん)
ISR セッションの数がしきい値を超えています。

IRR_使い果たされた
ISR セッションの数が、ノードに指定された最大数に達しました。

追加エラーが発生します
エンドポイント・セッションの数が指定された最大数に達しました。

AP_静止中
タイプ **静止の解除** または **静止** - ISR の STOP_NODE が発行されました。

cos_node_row_data.node_row.maximum.rar
0-255 範囲内の、追加の抵抗最大値を経路指定します。

cos_node_row_data.node_row.maximum.status
ノードの最大輻輳状況を指定します。このパラメーターは、**輻輳しています**されているその他の値のいずれかに、または論理 **それとも**を使用して結合された 2 つ以上のその他の値に設定することができます。可能な値は次のとおりです

輻輳しています
ISR セッションの数が、ノードの構成の *isr_sessions_upper_threshold* 値よりも低くなっています。

輻輳 (しげん)
ISR セッションの数がしきい値を超えています。

IRR_使い果たされた
ISR セッションの数が、ノードに指定された最大数に達しました。

追加エラーが発生します
エンドポイント・セッションの数が指定された最大数に達しました。

AP_静止中
タイプ **静止の解除** または **静止** - ISR の STOP_NODE が発行されました。

cos_node_row_data.node_row.weight
このノード行に関連付けられているウェイト。

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

無効なファイル名の変更
list_options パラメーターが **リストを含む** （ **包括的** ）に設定されました。指定された名前から始まるすべての項目をリストしますが、 *cos_name* パラメーターが無効でした。

ファイルの追加リスト・ オプション
list_options パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

QUERY_COS_TG_ROW

QUERY_COS_TG_ROW returns TG row information for a specified class of service as previously defined by DEFINE_COS (or implicitly by the node for the SNA-defined COSs).

This verb can be used to obtain information about a specific COS TG row, or about multiple COS TG rows, depending on the options used.

VCB 構造体

```
typedef struct query_cos_tg_row
{
    AP_UINT16           opcode;             /* verb operation code         */
    unsigned char       reserv2;            /* reserved                    */
    unsigned char       format;             /* reserved                    */
    AP_UINT16           primary_rc;         /* primary return code         */
    AP_UINT32           secondary_rc;       /* secondary return code       */
    unsigned char       *buf_ptr;           /* pointer to buffer           */
    AP_UINT32           buf_size;           /* buffer size                 */
    AP_UINT32           total_buf_size;     /* total buffer size required  */
    AP_UINT16           num_entries;        /* number of entries           */
    AP_UINT16           total_num_entries;  /* total number of entries     */
    unsigned char       list_options;       /* listing options             */
    unsigned char       reserv3;            /* reserved                    */
    unsigned char       cos_name[8];        /* cos name                    */
    AP_UINT16           tg_row_index;       /* TG row index                */
} QUERY_COS_TG_ROW;
```

```
typedef struct cos_tg_row_data
{
    AP_UINT16           overlay_size;       /* size of returned entry      */
    unsigned char       cos_name[8];        /* cos name                    */
    AP_UINT16           tg_row_index;       /* TG row index                */
    COS_TG_ROW          tg_row;             /* TG row information          */
} COS_TG_ROW_DATA;
```

```
typedef struct cos_tg_row
{
    TG_DEFINED_CHARS    minimum;            /* minimum                     */
    TG_DEFINED_CHARS    maximum;            /* maximum                     */
    unsigned char       weight;             /* weight                      */
    unsigned char       reserv1;            /* reserved                    */
} COS_TG_ROW;
```

```
typedef struct tg_defined_chars
{
    unsigned char       effect_cap;          /* Effective capacity          */
    unsigned char       reserve1[5];         /* Reserved                    */
    unsigned char       connect_cost;        /* Connection Cost             */
    unsigned char       byte_cost;           /* Byte cost                   */
    unsigned char       reserve2;            /* Reserved                    */
    unsigned char       security;            /* Security                    */
    unsigned char       prop_delay;          /* Propagation delay           */
    unsigned char       modem_class;         /* reserved                    */
    unsigned char       user_def_parm_1;     /* User-defined parameter 1    */
    unsigned char       user_def_parm_2;     /* User-defined parameter 2    */
    unsigned char       user_def_parm_3;     /* User-defined parameter 3    */
} TG_DEFINED_CHARS;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
文字列の追加 （ TG_TG_ROW ）

buf_ptr
CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

num_entries
データが戻される COS TG 行の最大数。 範囲ではなく、特定の COS TG 行のデータを要求するには、1 という値を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータを返すために開始する、 COS TG 行のリスト内の位置。 可能な値は次のとおりです

リストの最初のリスト （ _R ）
リストの最初の項目から開始します。

リストを含む （ 包括的 ）
cos_name パラメーターと *tg_row_index* パラメーターの組み合わせによって指定されたエントリーから開始します。

次への AP_LIST_FROM_NEXT
cos_name パラメーターと *tg_row_index* パラメーターの組み合わせによって指定されたエントリーの直後のエントリーから開始します。

このリストは、 *cos_name* によって配列され、その後 COS ごとに *tg_row_index* によって配列されます。 リストの順序とアプリケーションでの特定のエントリーの取得方法について詳しくは、34 ページの『List options for QUERY_* Verbs』を参照してください。

cos_name
データが必要なサービス・クラス名、またはリスト内の索引として使用される名前。 この名前は、 8 バイトのタイプ A の EBCDIC ストリング (文字で始まる) で、名前が 8 文字より短い場合は、右側に EBCDIC のスペースが埋め込まれます。 *list_options* を **リストの最初のリスト** （ _R ）に設定すると、このパラメーターは無視されます。

tg_row_index
データが必要な TG 行番号、またはリストへの索引として使用される番号 (最初の行の索引がゼロの場合)。 *list_options* を **リストの最初のリスト** （ _R ）に設定すると、このパラメーターは無視されます。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

cos_tg_row_data.overlay_size
The size of the returned `cos_tg_row_data` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `cos_tg_row_data` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

cos_tg_row_data.cos_name
Class of service name. This is an 8-byte type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces if the name is shorter than 8 characters.

cos_tg_row_data.tg_row_index
TG row index (the first row has an index of zero).

cos_tg_row_data.tg_row.minimum.effect_cap
Minimum limit for actual bits per second rate (line speed). The value is encoded as a 1-byte floating point number, represented by the formula 0.1 mmm * 2^{eeeee} where the bit representation of the byte is b'eeeeemmm'. Each unit of effective capacity is equal to 300 bits per second.

cos_tg_row_data.tg_row.minimum.connect_cost
Minimum limit for cost per connect time; an integer value in the range `0`-255, where `0` is the lowest cost per connect time and 255 is the highest.

cos_tg_row_data.tg_row.minimum.byte_cost
Minimum limit for cost per byte; an integer value in the range `0`-255, where zero is the lowest cost per byte and 255 is the highest.

cos_tg_row_data.tg_row.minimum.security
Minimum level of security. Possible values are:

AP_SEC_NONSECURE
No security.

AP_SEC_PUBLIC_SWITCHED_NETWORK
Data is transmitted over a public switched network.

AP_SEC_UNDERGROUND_CABLE
Data is transmitted over secure underground cable.

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

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

AP_SEC_ENCRYPTED
Data is encrypted before transmission over the line.

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

cos_tg_row_data.tg_row.minimum.prop_delay
Minimum limits for propagation delay: the time that a signal takes to travel the length of the link. Possible values are:

AP_PROP_DELAY_MINIMUM
Minimum propagation delay.

AP_PROP_DELAY_LAN
Delay is less than 480 microseconds (typical for a LAN). If the verb was issued to a running node, this value will be returned if the DEFINE_COS specified either AP_PROP_DELAY_LAN or AP_PROP_DELAY_MINIMUM.

AP_PROP_DELAY_TELEPHONE
Delay is in the range 480-49,512 microseconds (typical for a telephone network).

AP_PROP_DELAY_PKT_SWITCHED_NET
Delay is in the range 49,512-245,760 microseconds (typical for a packet-switched network).

AP_PROP_DELAY_SATELLITE
Delay is greater than 245,760 microseconds (typical for a satellite link).

AP_PROP_DELAY_MAXIMUM
Maximum propagation delay.

cos_tg_row_data.tg_row.minimum.user_def_parm_1 **through** ***cos_tg_row_data.tg_row.minimum.user_def_parm_3***
Minimum values for user-defined parameters, which include other TG characteristics not covered by the above parameters. Each of these parameters is set to a value in the range 0-255.

cos_tg_row_data.tg_row.maximum.effect_cap
Maximum limit for actual bits per second rate (line speed). The value is encoded as a 1-byte floating point number, represented by the formula 0.1 mmm * 2 ^{eeeee} where the bit representation of the byte is eeeeemmm. Each unit of effective capacity is equal to 300 bits per second.

cos_tg_row_data.tg_row.maximum.connect_cost
Maximum limit for cost per connect time; an integer value in the range 0-255, where 0 is the lowest cost per connect time and 255 is the highest.

cos_tg_row_data.tg_row.maximum.byte_cost
Maximum limit for cost per byte; an integer value in the range 0-255, where 0 is the lowest cost per byte and 255 is the highest.

cos_tg_row_data.tg_row.maximum.security
Maximum level of security. Possible values are:

AP_SEC_NONSECURE
No security.

AP_SEC_PUBLIC_SWITCHED_NETWORK
Data is transmitted over a public switched network.

AP_SEC_UNDERGROUND_CABLE
Data is transmitted over secure underground cable.

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

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

AP_SEC_ENCRYPTED
Data is encrypted before transmission over the line.

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

AP_SEC_MAXIMUM
Maximum security.

cos_tg_row_data.tg_row.maximum.prop_delay
Maximum limits for propagation delay: the time that a signal takes to travel the length of the link. Possible values are:

AP_PROP_DELAY_MINIMUM
Minimum propagation delay.

AP_PROP_DELAY_LAN
Delay is less than 480 microseconds (typical for a LAN).

AP_PROP_DELAY_TELEPHONE
Delay is in the range 480-49,512 microseconds (typical for a telephone network).

AP_PROP_DELAY_PKT_SWITCHED_NET
Delay is in the range 49,512-245,760 microseconds (typical for a packet-switched network).

AP_PROP_DELAY_SATELLITE
Delay is greater than 245,760 microseconds (typical for a satellite link). If the verb was issued to a running node, this value will be returned if the DEFINE_COS specified either AP_PROP_DELAY_SATELLITE or AP_PROP_DELAY_MAXIMUM.

AP_PROP_DELAY_MAXIMUM
Maximum propagation delay.

cos_tg_row_data.tg_row.maximum.user_def_parm_1* through *cos_tg_row_data.tg_row.maximum.user_def_parm_3
Maximum values for user-defined parameters, which include other TG characteristics not covered by the above parameters. Each of these parameters is set to a value in the range 0-255.

cos_tg_row_data.tg_row.weight
Weight associated with this TG row.

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_COS_NAME
The *list_options* parameter was set to AP_LIST_INCLUSIVE, to list all entries starting from the supplied name, but the *cos_name* parameter was not valid.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

照会 CPIC_SIDE_INFO

この verb は、指定されたシンボリック宛先名のサイド情報エントリー、または使用されるオプションに応じて、複数のシンボリック宛先名についてサイド情報エントリーを戻します。

この verb と CPI-C 関数 外部 CPIC_Side_Information の抽出との違いに注意してください。 この verb は、すべての CS Linux CPI-C アプリケーションによって使用されるデフォルト情報を戻すように構成ファイルを照会します。 CPI-C 関数は、アプリケーション自身のコピーをサイド情報テーブルのメモリー

内で照会します。サイド情報テーブルは、他の CPI-C サイド情報関数を使用してアプリケーションが変更した可能性があります。

この verb は、ドメイン構成ファイルに対して発行する必要があります。

VCB 構造体

```
typedef struct query_cpic_side_info
{
   AP_UINT16            opcode;              /* verb operation code         */
   unsigned char        reserv2;             /* reserved                    */
   unsigned char        format;              /* reserved                    */
   AP_UINT16            primary_rc;          /* primary return code         */
   AP_UINT32            secondary_rc;        /* secondary return code       */
   unsigned char        *buf_ptr;            /* pointer to buffer           */
   AP_UINT32            buf_size;            /* buffer size                 */
   AP_UINT32            total_buf_size;      /* total buffer size required  */
   AP_UINT16            num_entries;         /* number of entries           */
   AP_UINT16            total_num_entries;   /* total number of entries     */
   unsigned char        list_options;        /* listing options             */
   unsigned char        reserv3;             /* reserved                    */
   unsigned char        sym_dest_name[8];    /* Symbolic destination name   */
} QUERY_CPIC_SIDE_INFO;
```

```
typedef struct cpic_side_info_data
{
   AP_UINT16                overlay_size;     /* size of returned entry     */
   unsigned char            sym_dest_name[8]; /* Symbolic destination name  */
   unsigned char            reserv1[2];       /* reserved                   */
   CPIC_SIDE_INFO_DEF_DATA  def_data;
} CPIC_SIDE_INFO_DATA;
```

```
typedef struct cpic_side_info_def_data
{
   unsigned char            description[32];  /* resource description       */
   unsigned char            reserv1[16];      /* reserved                   */
   CPIC_SIDE_INFO           side_info;        /* CPIC side info             */
   unsigned char            user_data[24];    /* reserved                   */
} CPIC_SIDE_INFO_DEF_DATA;
```

```
typedef struct cpic_side_info
{
   unsigned char   partner_lu_name[17];       /* Fully qualified partner    */
                                              /*  LU name                   */
   unsigned char   reserved[3];               /* Reserved                   */
   AP_UINT32       tp_name_type;              /* TP name type               */
   unsigned char   tp_name[64];               /* TP name                    */
   unsigned char   mode_name[8];              /* Mode name                  */
   AP_UINT32       conversation_security_type; /* Conversation security     */
                                              /*  type                      */
   unsigned char   security_user_id[10];      /* User ID                    */
   unsigned char   security_password[10];     /* Password                   */
   unsigned char   lu_alias[8];               /* LU alias                   */
} CPIC_SIDE_INFO;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_CPIC_SIDE_INFO

buf_ptr
A pointer to a data buffer that CS Linux will use to return the requested information.

buf_size
Size of the supplied data buffer.

num_entries
Maximum number of symbolic destination names for which data should be returned. To request data for a specific symbolic destination name rather than a range, specify the value 1. To return as many

entries as possible, specify zero; in this case, CS Linux will return the maximum number of entries that can be accommodated in the supplied data buffer.

list_options
The position in the list of symbolic destination names from which CS Linux should begin to return data. Possible values are:

AP_FIRST_IN_LIST
Start at the first entry in the list.

AP_LIST_INCLUSIVE
Start at the entry specified by the *sym_dest_name* parameter.

AP_LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *sym_dest_name* parameter.

For more information about how the list is ordered and how the application can obtain specific entries from it, see "List options for QUERY_* Verbs" on page 34.

sym_dest_name
Symbolic destination 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 AP_FIRST_IN_LIST. The name is an ASCII string, consisting of uppercase A-Z and numerals 0-9, padded on the right with spaces if the name is shorter than 8 characters.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

cpic_side_info_data.overlay_size
The size of the returned `cpic_side_info_data` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `cpic_side_info_data` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

cpic_side_info_data.sym_dest_name
Symbolic destination name for the returned side information entry.

cpic_side_info_data.def_data.description
A null-terminated text string describing the side information entry, as specified in the definition of the side information entry.

cpic_side_info_data.def_data.side_info.partner_lu_name
Fully qualified name of the partner LU. The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

cpic_side_info_data.def_data.side_info.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:

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

XC_SNA_SERVICE_TP
Service TP. The TP name must be specified as an 8-character ASCII string representing the hexadecimal digits of a 4-character name. For example, if the hexadecimal representation of the name is 0x21F0F0F8, set the *def_data.side_info.tp_name* parameter to the 8-character string `21F0F0F8'.

The first character (represented by two bytes) must be a hexadecimal value in the range 0x0-0x3F, excluding 0x0E and 0x0F; the remaining characters (each represented by two bytes) must be valid EBCDIC characters.

cpic_side_info_data.def_data.side_info.tp_name
TP name of the target TP. This is a 64-byte ASCII character string, right-padded with spaces.

cpic_side_info_data.def_data.side_info.mode_name
Name of the mode used to access the target TP. This is an 8-byte ASCII character string, right-padded with spaces.

cpic_side_info_data.def_data.side_info.conversation_security_type
Specifies whether the target TP uses conversation security. Possible values are:

XC_SECURITY_NONE
The target TP does not use conversation security.

XC_SECURITY_PROGRAM
The target TP uses conversation security. The *security_user_id* and *security_password* parameters specified below will be used to access the target TP.

XC_SECURITY_PROGRAM_STRONG
As for XC_SECURITY_PROGRAM, except that the local node must not send the password across the network in clear text format. This value can be used only if the remote system supports password substitution.

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

cpic_side_info_data.def_data.side_info.security_user_id
User ID used to access the partner TP. This parameter is not used if the *conversation_security_type* parameter is set to XC_SECURITY_NONE.

cpic_side_info_data.def_data.side_info.security_password
Password used to access the partner TP. This parameter is used only if the *conversation_security_type* parameter is set to XC_SECURITY_PROGRAM or XC_SECURITY_PROGRAM_STRONG.

cpic_side_info_data.def_data.side_info.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.

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc

ファイルの追加リスト・オプション
list_options パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: state check

If the verb does not execute because of a state error, CS Linux returns the following parameters.

primary_rc
AP_STATE_CHECK

secondary_rc

AP_INVALID_SYM_DEST_NAME
The *list_options* parameter was set to AP_LIST_INCLUSIVE, to list all entries starting from the supplied name, but the *sym_dest_name* parameter was not valid.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_STATE_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

QUERY_CS_TRACE

This verb returns information about the current tracing options for data sent between computers on the CS Linux LAN. For more information about tracing options, see the *IBM Communications Server for Data Center Deployment on Linux Administration Guide*.

This verb must be issued to a running node.

VCB structure

```
typedef struct query_cs_trace
{
   AP_UINT16        opcode;            /* verb operation code               */
   unsigned char    reserv2;           /* reserved                          */
   unsigned char    format;            /* reserved                          */
   AP_UINT16        primary_rc;        /* primary return code               */
   AP_UINT32        secondary_rc;      /* secondary return code             */
   unsigned char    dest_sys[128];     /* node to which messages are traced  */
   unsigned char    reserv4[4];        /* reserved                          */
   AP_UINT16        trace_flags;       /* trace flags                       */
   AP_UINT16        trace_direction;   /* direction (send/rcv/both) to trace */
   unsigned char    reserv3[8];        /* Reserved                          */
} QUERY_CS_TRACE;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
照会の照会のトレース

dest_sys
トレース・オプションが照会されているサーバー名。 これは ASCII ストリングで、名前が 128 文字より短い場合は、右側にスペースが埋め込まれます。

この verb が発行されたコンピューター (NOF API 呼び出しの *target_handle* パラメーターによって識別される) と LAN 上の他の 1 つのサーバーの間で流れるメッセージのトレース・オプションを照会するには、他のサーバーの名前をここに指定します。

コンピューター名に . (ピリオド) 文字が含まれている場合、 CS Linux は、それが完全修飾名であると見なします。それ以外の場合は、 DNS ルックアップを実行してコンピューター名を判別します。

デフォルトのトレース・オプション (宛先システム 名が指定されていない SET_CS_TRACE verb によって設定される) を照会するには、このパラメーターをすべての ASCII スペース文字に設定します。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

secondary_rc
Not used.

trace_flags
The types of tracing currently active. For more information about these trace types, see "SET_CS_TRACE" on page 568.

If no tracing is active, or if tracing of all types is active, this is one of the following values:

AP_NO_TRACE
No tracing.

AP_ALL_TRACE
Tracing of all types.

If tracing is being used on specific interfaces, this parameter is set to one or more values from the list below, combined using a logical OR operation.

AP_CS_ADMIN_MSG
Internal messages relating to client/server topology

AP_CS_DATAGRAM
Datagram messages

AP_CS_DATA
Data messages

trace_direction
Specifies the direction or directions in which tracing is active. This parameter is not used if *trace_flags* is set to AP_NO_TRACE. Possible values are:

AP_CS_SEND
Messages flowing from the target computer to the computer defined by *dest_sys* are traced.

AP_CS_RECEIVE
Messages flowing from the computer defined by *dest_sys* to the target computer are traced.

AP_CS_BOTH
Messages flowing in both directions are traced.

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

追加の名前が見つかりません
dest_sys パラメーターで指定されたサーバーが存在しなかったか、または開始されませんでした。

追加ローカル・システム
dest_sys パラメーターによって指定されたサーバーは、この verb が発行されたターゲット・ノードと同じです。

ファイルの検証ターゲット
この verb は、スタンドアロン・サーバーで発行されました。 この verb は、クライアント/サーバー・システム上でのみ発行できます。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

QUERY_DEFAULT_PU

QUERY_DEFAULT_PU allows the user to query the default PU (defined using DEFINE_DEFAULT_PU).

VCB structure

```
typedef struct query_default_pu
{
   AP_UINT16        opcode;            /* verb operation code               */
   unsigned char    reserv2;           /* reserved                          */
   unsigned char    format;            /* reserved                          */
   AP_UINT16        primary_rc;        /* primary return code               */
   AP_UINT32        secondary_rc;      /* secondary return code             */
   unsigned char    def_pu_name[8];    /* default PU name                   */
   unsigned char    description[32];   /* resource description              */
   unsigned char    reserv1[16];       /* reserved                          */
   unsigned char    def_pu_sess[8];    /* PU name of active default session  */
   unsigned char    reserv3[16];       /* reserved                          */
} QUERY_DEFAULT_PU;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
照会のデフォルトの T_PU

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

***pu_name* の定義**
最新の DEFINE_DEFAULT_PU verb で指定された PU の名前。 これは、 8 バイトのタイプ A の EBCDIC ストリング (文字で始まる) で、右側に EBCDIC のスペースが埋め込まれます。 このフィールドがすべて 2 進ゼロに設定されている場合、これは、 DEFINE_DEFAULT_PU verb が発行されていないか、または プール名 パラメーターをすべてゼロとして指定した DEFINE_DEFAULT_PU verb を発行してデフォルト PU が削除されたことを示します。

記述
デフォルト PU の定義に指定されている、デフォルト PU を記述するヌル終了のテキスト・ストリング。

def_pu_sess
現在アクティブなデフォルト PU セッションに関連付けられている PU の名前。

このパラメーターには、通常、 *pu_name* の定義 フィールドと同じ値が含まれます。 ただし、デフォルト PU が定義されていても、その PU に関連付けられているセッションがアクティブでない場合、CS Linux は、定義されたデフォルト PU に関連付けられたセッションがアクティブになるまで、以前のデフォルト PU に関連付けられたセッションを引き続き使用します。 この場合、このパラメーターは前のデフォルト PU の名前を指定し、 *pu_name* の定義 フィールドとは異なります。

アクティブな PU セッションがない場合、このフィールドはすべて 2 進ゼロに設定されます。

Returned parameters: node not started

If the verb does not execute because the node has not yet been started, CS Linux returns the following parameters:

primary_rc
AP_NODE_NOT_STARTED

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

QUERY_DEFAULTS

QUERY_DEFAULTS allows the user to query the default parameters defined for the node (defined using DEFINE_DEFAULTS).

VCB structure

```
typedef struct query_defaults
{
     AP_UINT16          opcode;             /* verb operation code            */
     unsigned char      reserv2;            /* reserved                       */
     unsigned char      format;             /* reserved                       */
     AP_UINT16          primary_rc;         /* primary return code            */
     AP_UINT32          secondary_rc;       /* secondary return code          */
     DEFAULT_CHARS      default_chars;      /* default parameters             */
} QUERY_DEFAULTS;
```

```
typedef struct default_chars
{
     unsigned char      description[32];    /* resource description           */
     unsigned char      reserv2[16];        /* reserved                       */
     unsigned char      mode_name[8];       /* default mode name              */
     unsigned char      implicit_plu_forbidden; /* disallow implicit PLUs?    */
     unsigned char      specific_security_codes;/*generic security sensecodes?*/
     AP_UINT16          limited_timeout;    /* timeout for limited sessions   */
     unsigned char      reserv[244];        /* reserved                       */
} DEFAULT_CHARS;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
照会のデフォルト

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

default_chars.description
DEFINE_DEFAULTS で指定されている、デフォルト・パラメーターを記述するヌル終了のテキスト・ストリング。

デフォルト t_chars.mode_name
デフォルト・モードの名前。 セッションを開始しようとするときに、認識されないモード名をアプリケーションが指定すると、このモードからのパラメーターが、認識されないモードのデフォルト定義として使用されます。

モード名は、 8 バイトのタイプ A の EBCDIC ストリングです。 DEFINE_DEFAULTS verb を使用してデフォルト・モード名が指定されていない場合、このパラメーターは 8 個の 2 進ゼロに設定されます。

default_chars.implicit_plu_禁じられた暗黙
CS Linux が不明なパートナー LU の代わりに暗黙定義を配置するかどうかを示します。 可能な値は次のとおりです

類人猿
CS Linux は、不明なパートナー LU の代わりに暗黙定義を配置しません。 すべてのパートナー LU が明示的に定義されている必要

アプ・ノー
CS Linux は、不明なパートナー LU に代わって暗黙定義を代わりに使用します。

デフォルトの文字数 . 特殊セキュリティー・コード
CS Linux がセキュリティー認証または許可の失敗で特定のセンス・コードを使用するかどうかを示します。 特定のセンス・コードが戻されるのは、セッションでそれらのパートナー LU に対してサポートを報告しているパートナー LU のみです。 可能な値は次のとおりです

類人猿
CS Linux では特定のセンス・コードを使用

アプ・ノー
CS Linux は特定のセンス・コードを使用しません。

default_chars.limited_timeout
フリーの限定リソース・コン勝者セッションが非活動化されるまでのタイムアウトを指定します。 範囲は 0-65,535 秒です。

戻りパラメーター: ノードが開始していません

ノードがまだ開始されていないために verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
追加ノードが開始されました

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

QUERY_DIRECTORY_ENTRY

QUERY_DIRECTORY_ENTRY は、ディレクトリー・データベース内のリソースに関する情報を戻します。 使用されるオプションに応じて、特定のリソースまたは複数のリソースに関する要約情報または詳細情報のいずれかを戻すことができます。

実行中のノードに対して verb が発行された場合は、明示的に定義されたリソース (DEFINE_DIRECTORY_ENTRY、または DEFINE_ADJACENT_LEN_NODE を使用) と、動的に検出されたリソースに関する情報を両方とも戻します。 ノードが稼働していない場合は、明示的に定義されたエントリーのみが戻されます。

verb がエンド・ノードに発行されると、そのディレクトリーにはエンド・ノードとそのリソースに関する情報だけが含まれ、他のノードについての情報は含まれません。 最初に戻される項目は、エンド・ノード

自体に対するもので、その後に LU が続きます。 (エンド・ノードのネットワーク・ノード・サーバーについては、エントリーは戻されません。)

ネットワーク・ノードに対して verb が発行されると、ディレクトリーには、複数のネットワーク・ノードとそれに関連するエンド・ノードおよび LU に関する情報が含まれることがあり 各ネットワーク・ノードについて、戻される情報は以下の順序になっています。

1. ネットワーク・ノード。
2. このノードが所有する LU 。
3. ネットワーク・ノードに関連付けられた最初のエンド・ノード。
4. このエンド・ノードが所有する LU 。
5. ネットワーク・ノードに関連付けられている他のすべてのエンド・ノード。各ノードの後に LU が続き

VCB structure

```
typedef struct query_directory_entry
{
   AP_UINT16         opcode;               /* verb operation code            */
   unsigned char     reserv2;              /* reserved                       */
   unsigned char     format;               /* reserved                       */
   AP_UINT16         primary_rc;           /* primary return code            */
   AP_UINT32         secondary_rc;         /* secondary return code          */
   unsigned char     *buf_ptr;             /* pointer to buffer              */
   AP_UINT32         buf_size;             /* buffer size                    */
   AP_UINT32         total_buf_size;       /* total buffer size required     */
   AP_UINT16         num_entries;          /* number of entries              */
   AP_UINT16         total_num_entries;    /* total number of entries        */
   unsigned char     list_options;         /* listing options                */
   unsigned char     reserv3;              /* reserved                       */
   unsigned char     resource_name[17];    /* network qualified resource     */
                                           /* name                           */
   unsigned char     reserv4;              /* reserved                       */
   AP_UINT16         resource_type;        /* Resource type                  */
   unsigned char     parent_name[17];      /* parent name filter             */
   unsigned char     reserv5;              /* reserved                       */
   AP_UINT16         parent_type;          /* parent type                    */
   unsigned char     reserv6[24];          /* reserved                       */
} QUERY_DIRECTORY_ENTRY;
```

```
typedef struct directory_entry_summary
{
   AP_UINT16         overlay_size;          /* size of this entry            */
   unsigned char     resource_name[17];     /* network qualified resource    */
                                            /* name                          */
   unsigned char     reserve1;              /* reserved                      */
   AP_UINT16         resource_type;         /* Resource type                 */
   unsigned char     description[32];       /* resource description          */
   unsigned char     reserv1[16];           /* reserved                      */
   AP_UINT16         real_owning_cp_type;     /* CP type of real owner       */
   unsigned char     real_owning_cp_name[17]; /* CP name of real owner       */
   unsigned char     reserve2;                /* reserved                    */
} DIRECTORY_ENTRY_SUMMARY;
```

```
typedef struct directory_entry_detail
{
   AP_UINT16         overlay_size;          /* size of this entry            */
   unsigned char     resource_name[17];     /* network qualified res name    */
   unsigned char     reserv1a;              /* reserved                      */
   AP_UINT16         resource_type;         /* Resource type                 */
   unsigned char     description[32];       /* resource description          */
   unsigned char     reserv2[16];           /* reserved                      */
   unsigned char     parent_name[17];       /* Network qualified parent name */
   unsigned char     reserv1b;              /* reserved                      */
   AP_UINT16         parent_type;           /* Parent resource type          */
   unsigned char     entry_type;            /* Type of the directory entry   */
   unsigned char     location;              /* Resource location             */
   AP_UINT16         real_owning_cp_type;     /* CP type of real owner       */
   unsigned char     real_owning_cp_name[17]; /* CP name of real owner       */
   unsigned char     reserv1c;              /* reserved                      */
   AP_UINT16         supplier_cp_type;        /* CP type of supplier        */
   unsigned char     supplier_cp_name[17];    /* CP name of supplier        */
```

```
   unsigned char     reserva;              /* reserved                     */
} DIRECTORY_ENTRY_DETAIL;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
エントリーの追加 （ _QUERY_DIRECTORY_ENTRY ）

buf_ptr
CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

num_entries
データが戻されるリソースの最大数。 特定の範囲ではなく、特定のリソースのデータを要求するには、 1 という値を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、 CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータの戻りを開始する必要のあるリスト内の位置、および各項目に必要な情報のレベル。 以下の値のいずれかを使用して、情報のレベルを指定します。

要約の要約
サマリー情報のみ。

追加の詳細
詳細情報。

以下のいずれかの値を指定して、論理 **それとも** 操作を使用してこの値を結合します。

リストの最初のリスト （ _R ）
リストの最初の項目から開始します。

リストを含む （ 包括的 ）
parent_name パラメーター、 リソース名パラメーター、および リソース・タイプ パラメーターの組み合わせによって指定されたエントリーから開始します。

次への AP_LIST_FROM_NEXT
parent_name、 リソース名、および リソース・タイプ パラメーターの組み合わせによって指定されたエントリーの直後のエントリーから開始します。

このリストは、 *parent_name* によって、次に リソース名によって順序付けされ、最後に リソース・タイプによって配列されます。 リストの順序とアプリケーションでの特定のエントリーの取得方法について詳しくは、 34 ページの『List options for QUERY_* Verbs』を参照してください。

リソース名
情報が必要なリソースの完全修飾名、またはリソースのリストへの索引として使用される名前。 *list_options* を **リストの最初のリスト** （ _R ）に設定すると、この値は無視されます。

この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

リソース・タイプ
情報が必要なリソースのタイプ。 *list_options* を **リストの最初のリスト** （ _R ）に設定すると、この値は無視されます。 可能な値は次のとおりです

AP_ENCP_RESOURCE
エンド・ノードまたは LEN ノード

リソースの追加
ネットワーク・ノード

AP_LU_RESOURCE
ルゥ

parent_name
親リソースの完全修飾リソース名。 LU の場合、親リソースは所有制御点であり、エンド・ノードまたは LEN ノードの場合はネットワーク・ノード・サーバーになります。 指定された親に属するエントリーだけを返すには、このパラメーターを親リソースの名前に設定し、 *parent_type* を親のリソース・タイプに設定します。すべてのエントリーを戻すには、両方のパラメーターを 2 進ゼロに設定します。

この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、 1 から 8 文字の A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および 1 から 8 文字の A ストリング文字のネットワーク名で構成されます。

parent_type
親リソースのリソース・タイプ。 指定された親に属するエントリーのみを返すには、このパラメーターを親リソースのタイプに設定します。すべてのエントリーを戻すには、このパラメーターをゼロに設定します。 可能な値は次のとおりです

AP_ENCP_RESOURCE
エンド・ノード (エンド・ノードによって所有される LU リソースの場合)

リソースの追加
ネットワーク・ノード (ネットワーク・ノードが所有する LU リソースの場合、または EN リソースまたは LEN リソースの場合)

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

buf_size
提供されたバッファーに戻された情報の長さ。

total_buf_size
要求されたすべてのリスト情報を戻すために必要となるバッファーのサイズを示す戻り値。 *buf_size* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

num_entries
データ・バッファーに戻されたエントリーの数。

total_num_entries
使用可能な項目の合計数。 *num_entries* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

データ・バッファー内の各項目は、以下のパラメーターで構成されます。

ディレクトリー・エントリーの要約 . サイズがオーバーラップしています
戻された **ディレクトリー・ エントリーの要約** 構造体のサイズ。すなわち、データ・バッファー内の次のエントリーの先頭までのオフセット。

アプリケーションが戻されたバッファーを調べて、各 **ディレクトリー・ エントリーの要約** 構造体を順番に検出する必要がある場合は、この値を使用して次のデータ構造の正しいオフセットに移動しなければなりません。また、 C `sizeof()` 演算子を使用してはなりません。 これは、返されるオーバーレイのサイズが、 CS Linux の将来のリリースで増加する可能性があるためです。戻されたオーバーレイ・サイズを使用すると、アプリケーションが今後のリリースで引き続き動作することが保証されます。

ディレクトリー・エントリー・サマリー . リソース名
リソースの完全修飾名。 この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、 1 から 8 文字の A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および 1 から 8 文字の A ストリング文字のネットワーク名で構成されます。

ディレクトリー・エントリー・サマリー . リソース・タイプ
リソースのタイプ。 これは、以下のいずれかです。

AP_ENCP_RESOURCE
エンド・ノードまたは LEN ノード

リソースの追加
ネットワーク・ノード

AP_LU_RESOURCE
ルゥ

ディレクトリー・エントリーのサマリー . 説明
ディレクトリー項目を記述するヌル終了のテキスト・ストリング (ディレクトリー項目の定義に指定されているとおり) 。

ディレクトリー・サマリー . *real_owning_cp_type*
このパラメーターは、ローカル・ノードがネットワーク・ノードまたは分岐ネットワーク・ノードである場合にのみ適用されます。それ以外の場合は予約済みです。

このディレクトリー項目によって識別されるリソースを所有する実 CP が、親リソースであるか、別のノードであるかを指定します。 これは、以下のいずれかです。

追加なし
実際の所有者は、親リソースです。

AP_ENCP_RESOURCE
実際の所有者は、親リソースではないエンド・ノードです。 例えば、リソースが分岐ネットワーク・ノード (BrNN) のドメイン内のエンド・ノードによって所有されている場合、この BrNN のネットワーク・ノード・サーバーのディレクトリーには、親リソースとしての BrNN が含まれますが、実際の所有 CP はエンド・ノードです。

ディレクトリー・サマリー . 実所有者 *cp_name*
このパラメーターは、ローカル・ノードがネットワーク・ノードまたは分岐ネットワーク・ノードである場合にのみ適用されます。それ以外の場合は予約済みです。

所有者 *cp_type* の再レルム パラメーターが、リソースの実所有者が親ではないことを示している場合、このパラメーターは、リソースを所有する CP の完全修飾名を指定します。それ以外の場合は、予約されます。

この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、 1 から 8 文字の A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および 1 から 8 文字の A ストリング文字のネットワーク名で構成されます。

ディレクトリー・エントリー詳細 . オーバーレイ・サイズ
戻された **ディレクトリー・ エントリーの詳細** 構造体のサイズ。すなわち、データ・バッファー内の次のエントリーの先頭までのオフセット。

アプリケーションが戻されたバッファーを調べて、各 **ディレクトリー・ エントリーの詳細** 構造体を順番に検出する必要がある場合は、この値を使用して次のデータ構造の正しいオフセットに移動しなければなりません。また、 C `sizeof()` 演算子を使用してはなりません。 これは、返されるオーバーレイのサイズが、 CS Linux の将来のリリースで増加する可能性があるためです。戻されたオーバーレイ・サイズを使用すると、アプリケーションが今後のリリースで引き続き動作することが保証されます。

directory_entry_detail.resource_name
リソースの完全修飾名。 この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、 1 から 8 文字の A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および 1 から 8 文字の A ストリング文字のネットワーク名で構成されます。

directory_entry_detail.resource_type
リソースのタイプ。 これは、以下のいずれかです。

AP_ENCP_RESOURCE
エンド・ノードまたは LEN ノード

リソースの追加
ネットワーク・ノード

AP_LU_RESOURCE
ルゥ

ディレクトリー・エントリー詳細 . 説明
ディレクトリー項目を記述するヌル終了のテキスト・ストリング (ディレクトリー項目の定義に指定されているとおり) 。

directory_entry_detail.parent_name
親リソースの完全修飾リソース名。 LU の場合、親リソースは所有制御点であり、エンド・ノードまたは LEN ノードの場合はネットワーク・ノード・サーバーになります。 このパラメーターはネットワーク・ノード・リソースには使用されません。

この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、 1 から 8 文字の A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および 1 から 8 文字の A ストリング文字のネットワーク名で構成されます。

ディレクトリー・エントリー・エントリー・タイプ . *parent_type*
親リソースのリソース・タイプ。 ネットワーク・ノード・リソースの場合、このパラメーターは使用されません。 それ以外の場合は、次のいずれかになります。

AP_ENCP_RESOURCE
エンド・ノード (エンド・ノードによって所有される LU リソースの場合)

リソースの追加
ネットワーク・ノード (ネットワーク・ノードが所有する LU リソースの場合、または EN リソースまたは LEN リソースの場合)

directory_entry_detail.entry_type
ディレクトリー項目のタイプを指定します。 これは、以下のいずれかです。

アプアホーム
ローカル・リソース。

AP_CACHE
キャッシュされた項目。

登録の登録
登録済みリソース (NN のみ)。

directory_entry_detail.location
リソースのロケーションを指定します。 これは、以下のいずれかです。

ローカル （ ローカル ）
リソースはローカル・ノードにあります。

ドメイン （ ドメイン ）
リソースは接続されたエンド・ノードに属しています。

追加クロスドメイン
リソースがローカル・ノードのドメイン内にありません。

ディレクトリー・エントリーの詳細 . *real_owning_cp_type*
このパラメーターは、ローカル・ノードがネットワーク・ノードまたは分岐ネットワーク・ノードである場合にのみ適用されます。それ以外の場合は予約済みです。

このディレクトリー項目によって識別されるリソースを所有する実 CP が、親リソースであるか、別のノードであるかを指定します。 これは、以下のいずれかです。

追加なし
実際の所有者は、親リソースです。

AP_ENCP_RESOURCE
実際の所有者は、親リソースではないエンド・ノードです。 例えば、リソースが分岐ネットワーク・ノード (BrNN) のドメイン内のエンド・ノードによって所有されている場合、この BrNN のネットワーク・ノード・サーバーのディレクトリーには、親リソースとしての BrNN が含まれますが、実際の所有 CP はエンド・ノードです。

directory_entry_detail.real_owning_cp_name
このパラメーターは、ローカル・ノードがネットワーク・ノードまたは分岐ネットワーク・ノードである場合にのみ適用されます。それ以外の場合は予約済みです。

所有者 *cp_type* の再レルム パラメーターが、リソースの実所有者が親ではないことを示している場合、このパラメーターは、リソースを所有する CP の完全修飾名を指定します。それ以外の場合は、予約されます。

この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、 1 から 8 文字の A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および 1 から 8 文字の A ストリング文字のネットワーク名で構成されます。

directory_entry_detail. サプライ cp_type
このパラメーターは、ローカル・ノードがネットワーク・ノードまたは分岐ネットワーク・ノードである場合にのみ適用されます。それ以外の場合は予約済みです。

このディレクトリー項目が、リソースの所有 CP ではない別のノードによって登録されたかどうかを指定します。 これは、以下のいずれかです。

追加なし
ディレクトリー項目が登録されていないか、または所有する CP によって登録されています。

AP_ENCP_RESOURCE
ディレクトリー項目は、所有している CP ではないノードによって登録されました。 例えば、リソースが、ローカル・ノードのドメイン内にある分岐ネットワーク・ノード (BrNN) のドメイン内のエンド・ノードによって所有されている場合、 BrNN はそのリソースをローカル・ノードに登録するため、サプライヤーです。ただし、実際の所有 CP はエンド・ノードです。

directory_entry_detail.cpier_cp_name
このパラメーターは、ローカル・ノードがネットワーク・ノードまたは分岐ネットワーク・ノードである場合にのみ適用されます。それ以外の場合は予約済みです。

サプライ *cp_type* パラメーターが、ディレクトリー項目が所有リソースではないノードによって登録されたことを示している場合、このパラメーターは、登録を提供した CP の完全修飾名を指定します。それ以外の場合は、予約されます。

この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、 1 から 8 文字の A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および 1 から 8 文字の A ストリング文字のネットワーク名で構成されます。

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_RES_NAME
The *list_options* parameter was set to AP_LIST_INCLUSIVE, to list all entries starting from the supplied name, but the *resource_name* parameter was not valid.

AP_INVALID_RES_TYPE
The *resouce_type* parameter was not set to a valid value.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

クエリー・ディレクトリー・ルー

QUERY_DIRECTORY_LU は、ディレクトリー・データベースから LU のリストを戻します。 これは、使用されるオプションに応じて、特定の LU または複数の LU に関する情報を取得するために使用できます。

この verb は実行中のノードに対して発行する必要があります

VCB structure

```
typedef struct query_directory_lu
{
   AP_UINT16         opcode;               /* verb operation code            */
   unsigned char     reserv2;              /* reserved                       */
   unsigned char     format;               /* reserved                       */
   AP_UINT16         primary_rc;           /* primary return code            */
   AP_UINT32         secondary_rc;         /* secondary return code          */
   unsigned char     *buf_ptr;             /* pointer to buffer              */
   AP_UINT32         buf_size;             /* buffer size                    */
   AP_UINT32         total_buf_size;       /* total buffer size required     */
   AP_UINT16         num_entries;          /* number of entries              */
   AP_UINT16         total_num_entries;    /* total number of entries        */
   unsigned char     list_options;         /* listing options                */
   unsigned char     reserv3;              /* reserved                       */
   unsigned char     lu_name[17];          /* network qualified lu name      */
} QUERY_DIRECTORY_LU;
```

```
typedef struct directory_lu_summary
{
   AP_UINT16         overlay_size;         /* size of returned entry         */
   unsigned char     lu_name[17];          /* network qualified lu name      */
   unsigned char     description[32];      /* resource description           */
   unsigned char     reserv1[16];          /* reserved                       */
} DIRECTORY_LU_SUMMARY;
```

```
typedef struct directory_lu_detail
{
   AP_UINT16         overlay_size;         /* size of returned entry          */
   unsigned char     lu_name[17];          /* network qualified lu name       */
   unsigned char     description[32];      /* resource description            */
   unsigned char     reserv1[16];          /* reserved                        */
   unsigned char     server_name[17];      /* network qualified server name   */
   unsigned char     lu_owner_name[17];    /* network qualified lu owner name */
   unsigned char     location;             /* Resource location               */
   unsigned char     entry_type;           /* Type of the directory entry     */
   unsigned char     wild_card;            /* type of wildcard entry          */
   unsigned char     apparent_lu_owner_name[17]; /* name of apparent LU owner */
   unsigned char     reserva[3];           /* reserved                        */
} DIRECTORY_LU_DETAIL;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
ファイルの追加 （ _L ）

buf_ptr
CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

num_entries
データが戻される LU の最大数。 特定の範囲ではなく、特定の LU のデータを要求するには、値 1 を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、 CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータの戻りを開始する必要のあるリスト内の位置、および各項目に必要な情報のレベル。 以下の値のいずれかを使用して、情報のレベルを指定します。

要約の要約
サマリー情報のみ。

追加の詳細
詳細情報。

以下のいずれかの値を指定して、論理 **それとも** 操作を使用してこの値を結合します。

リストの最初のリスト （ _R ）
リストの最初の項目から開始します。

リストを含む （ 包括的 ）
lu_name パラメーターで指定されたエントリーから開始します。

次への AP_LIST_FROM_NEXT
lu_name パラメーターによって指定されたエントリーの直後のエントリーから開始します。

リストの順序とアプリケーションでの特定のエントリーの取得方法について詳しくは、 34 ページの『List options for QUERY_* Verbs』を参照してください。

lu_name
情報が必要な LU の完全修飾名、または LU のリストへの索引として使用される名前。 *list_options* を **リストの最初のリスト** （ _R)に設定すると、この値は無視されます。

この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

buf_size
提供されたバッファーに戻された情報の長さ。

total_buf_size
要求されたすべてのリスト情報を戻すために必要となるバッファーのサイズを示す戻り値。 *buf_size* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

num_entries
データ・バッファーに戻されたエントリーの数。

total_num_entries
使用可能な項目の合計数。 *num_entries* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

データ・バッファー内の各項目は、以下のパラメーターで構成されます。

ディレクトリー・lu_summary.オーバーレイ・サイズ
戻された **ディレクトリー・ lu_summary** 構造体のサイズ。すなわち、データ・バッファー内の次のエントリーの先頭までのオフセット。

アプリケーションが戻されたバッファーを調べて、各 **ディレクトリー・ lu_summary** 構造体を順番に検出する必要がある場合は、この値を使用して次のデータ構造の正しいオフセットに移動しなければなりません。また、 C `sizeof()` 演算子を使用してはなりません。 これは、返されるオーバーレイ

のサイズが、 CS Linux の将来のリリースで増加する可能性があるためです。戻されたオーバーレイ・サイズを使用すると、アプリケーションが今後のリリースで引き続き動作することが保証されます。

ディレクトリー・エラー要約 . ***lu_name***
LU の完全修飾名。 この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、 1 から 8 文字の A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および 1 から 8 文字の A ストリング文字のネットワーク名で構成されます。

ディレクトリー・ ***lu_summary.description***
ディレクトリー項目を記述するヌル終了のテキスト・ストリング (ディレクトリー項目の定義に指定されているとおり) 。

directory_lu_detail.**オーバーレイ・サイズ**
戻された **ディレクトリー・** `lu_detail` 構造体のサイズ。すなわち、データ・バッファー内の次のエントリーの先頭までのオフセット。

アプリケーションが戻されたバッファーを調べて、各 **ディレクトリー・** `lu_detail` 構造体を順番に検出する必要がある場合は、この値を使用して次のデータ構造の正しいオフセットに移動しなければなりません。また、 C `sizeof()` 演算子を使用してはなりません。 これは、返されるオーバーレイのサイズが、 CS Linux の将来のリリースで増加する可能性があるためです。戻されたオーバーレイ・サイズを使用すると、アプリケーションが今後のリリースで引き続き動作することが保証されます。

ディレクトリー・ディレクトリー詳細 . ***lu_name***
LU の完全修飾名。 この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、 1 から 8 文字の A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および 1 から 8 文字の A ストリング文字のネットワーク名で構成されます。

directory_lu_detail.description
ディレクトリー項目を記述するヌル終了のテキスト・ストリング (ディレクトリー項目の定義に指定されているとおり) 。

directory_lu_detail.server_name
LU を提供するノードの完全修飾名。 この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、 1 から 8 文字の A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および 1 から 8 文字の A ストリング文字のネットワーク名で構成されます。

directory_lu_detail.lu_owner_name
LU を所有するノードの完全修飾名。 この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、 1 から 8 文字の A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および 1 から 8 文字の A ストリング文字のネットワーク名で構成されます。

directory_lu_detail.location
リソースのロケーションを指定します。 これは、以下のいずれかです。

ローカル (ローカル)
リソースはローカル・ノードにあります。

ドメイン (ドメイン)
リソースは接続されたエンド・ノードに属しています。

追加クロスドメイン
リソースがローカル・ノードのドメイン内にありません。

directory_lu_detail.entry_type
リソースのタイプを指定します。 これは、以下のいずれかです。

アプアホーム
ローカル・リソース。

AP_CACHE
キャッシュされた項目。

登録の登録
登録済みリソース (NN のみ)。

***directory_lu_detail.*ワイルドカード**
LU 項目が明示的な名前であるか、または名前の範囲に一致するワイルドカード値のものであるかを指定します。 これは、以下のいずれかです。

明示的
エントリーは明示的な LU 名です。

追加のワイルドカード （ _C ）
エントリーは、すべての LU 名に一致する完全なワイルドカード値です。

パーティション・ワイルドカードの追加
項目は部分ワイルドカードです。名前の中の非ブランク文字は、 LU 名との突き合わせに使用されます。

その他
LU 項目のタイプが不明です。

directory_lu_detail.*器具・ *lu_owner_name
このパラメーターは、ローカル・ノードがネットワーク・ノードまたは分岐ネットワーク・ノードである場合にのみ適用されます。それ以外の場合は予約済みです。

この LU の明白な所有する CP が実際の所有 CP ではない場合、このパラメーターは、明示的に所有する CP の完全修飾名を指定します。それ以外の場合は、予約されます。 例えば、リソースが分岐ネットワーク・ノード (BrNN) のドメイン内のエンド・ノードによって所有されている場合、この BrNN のネットワーク・ノード・サーバーのディレクトリーには、明らかな所有者として BrNN が含まれていますが、実際の所有者 CP が終了ノードです。

この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、 1 から 8 文字の A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および 1 から 8 文字の A ストリング文字のネットワーク名で構成されます。

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_LU_NAME
The *list_options* parameter was set to AP_LIST_INCLUSIVE, to list all entries starting from the supplied name, but the *lu_name* parameter was not valid.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

クエリー・ディレクトリー統計

QUERY_DIRECTORY_STATS は、ディレクトリー・データベース統計を戻します。これを使用して、ネットワーク・ロケーション・トラフィックのレベルを測定することができます。 ネットワーク・ノードの場合は、ディレクトリー・キャッシュの使用法に関する情報を戻します。この情報を使用して、DEFINE_NODE verb で指定された適切なキャッシュ・サイズを判別することができます。

この verb は実行中のノードに対して発行する必要があります

VCB 構造体

```
typedef struct query_directory_stats
{
   AP_UINT16        opcode;                 /* verb operation code         */
   unsigned char    reserv2;                /* reserved                    */
   unsigned char    format;                 /* reserved                    */
   AP_UINT16        primary_rc;             /* primary return code         */
   AP_UINT32        secondary_rc;           /* secondary return code       */
   AP_UINT32        max_caches;             /* maximum number of cache     */
                                            /*  entries                    */
   AP_UINT32        cur_caches;             /* cache entry count           */
   AP_UINT32        cur_home_entries;       /* home entry count            */
   AP_UINT32        cur_reg_entries;        /* registered entry count      */
   AP_UINT32        cur_directory_entries;  /* current number of directory */
                                            /*  entries                    */
   AP_UINT32        cache_hits;             /* count of cache finds        */
   AP_UINT32        cache_misses;           /* count of resources found    */
                                            /*  by broadcast search        */
                                            /*  (not in cache)             */
   AP_UINT32        in_locates;             /* locates in                  */
   AP_UINT32        in_bcast_locates;       /* broadcast locates in        */
   AP_UINT32        out_locates;            /* locates out                 */
   AP_UINT32        out_bcast_locates;      /* broadcast locates out       */
   AP_UINT32        not_found_locates;      /* unsuccessful locates        */
   AP_UINT32        not_found_bcast_locates;/* unsuccessful broadcast      */
                                            /*  locates                    */
   AP_UINT32        locates_outstanding;    /* total outstanding locates   */
   unsigned char    reserva[20];            /* reserved                    */
} QUERY_DIRECTORY_STATS;
```

Supplied parameters

The application supplies the following parameter:

opcode
AP_QUERY_DIRECTORY_STATS

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

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.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

QUERY_DLC

QUERY_DLC returns information about DLCs. This information is structured as "determined data" (data gathered dynamically during execution) and "defined data" (data supplied on DEFINE_DLC).

This verb can be used to obtain either summary or detailed information, about a specific DLC or about multiple DLCs, depending on the options used.

VCB structure

```
typedef struct query_dlc
{
   AP_UINT16       opcode;              /* verb operation code             */
   unsigned char   reserv2;             /* reserved                        */
   unsigned char   format;              /* reserved                        */
   AP_UINT16       primary_rc;          /* primary return code             */
   AP_UINT32       secondary_rc;        /* secondary return code           */
   unsigned char   *buf_ptr;            /* pointer to buffer               */
   AP_UINT32       buf_size;            /* buffer size                     */
   AP_UINT32       total_buf_size;      /* total buffer size required      */
   AP_UINT16       num_entries;         /* number of entries               */
   AP_UINT16       total_num_entries;   /* total number of entries         */
   unsigned char   list_options;        /* listing options                 */
   unsigned char   reserv3;             /* reserved                        */
   unsigned char   dlc_name[8];         /* name of DLC                     */
} QUERY_DLC;
```

```
typedef struct dlc_summary
{
   AP_UINT16       overlay_size;        /* size of returned entry          */
   unsigned char   dlc_name[8];         /* name of DLC                     */
   unsigned char   description[32];     /* resource description            */
   unsigned char   reserv1[16];         /* reserved                        */
   unsigned char   state;               /* State of the DLC                */
   unsigned char   dlc_type;            /* DLC type                        */
} DLC_SUMMARY;
```

```
typedef struct dlc_detail
{
   AP_UINT16       overlay_size;        /* size of returned entry          */
   unsigned char   dlc_name[8];         /* name of DLC                     */
```

```
   unsigned char   reserv2[2];          /* reserved                         */
   DLC_DET_DATA    det_data;            /* Determined data                  */
   DLC_DEF_DATA    def_data;            /* Defined data                     */
} DLC_DETAIL;
```

```
typedef struct dlc_det_data
{
   unsigned char   state;               /* State of the DLC                 */
   unsigned char   reserv3[3];          /* reserved                         */
   unsigned char   reserva[20];         /* reserved                         */
} DLC_DET_DATA;
```

```
typedef struct dlc_def_data
{
   unsigned char   description[32];      /* resource description            */
   unsigned char   initially_active;     /* is DLC initially active?        */
   unsigned char   reserv1[15];          /* reserved                        */
   unsigned char   dlc_type;             /* DLC type                        */
   unsigned char   neg_ls_supp;          /* negotiable link station support */
   unsigned char   port_types;           /* port types supported by DLC type */
   unsigned char   hpr_only;             /* only support HPR?               */
   unsigned char   reserv3;              /* reserved                        */
   unsigned char   retry_flags;          /* reserved                        */
   AP_UINT16       max_activation_attempts; /* reserved                     */
   AP_UINT16       activation_delay_timer;  /* reserved                     */
   unsigned char   reserv4[4];              /* reserved                     */
   AP_UINT16       dlc_spec_data_len;       /* Length of DLC specific data  */
} DLC_DEF_DATA;
```

For more details of the DLC-specific data, see “DEFINE_DLC” on page 77. The data structure for this data follows the `dlc_def_data` structure, but is padded to start on a 4-byte boundary.

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
追加の照会

buf_ptr
CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

num_entries
データが戻される DLC の最大数。 範囲ではなく、特定の DLC のデータを要求するには、 1 という値を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、 CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータの戻りを開始する必要のあるリスト内の位置、および各項目に必要な情報のレベル。 以下の値のいずれかを使用して、情報のレベルを指定します。

要約の要約
サマリー情報のみ。

追加の詳細
詳細情報。

以下のいずれかの値を指定して、論理 **それとも** 操作を使用してこの値を結合します。

リストの最初のリスト （ _R ）
リストの最初の項目から開始します。

リストを含む （ 包括的 ）
dlc_name パラメーターで指定されたエントリーから開始します。

次への AP_LIST_FROM_NEXT
dlc_name パラメーターによって指定されたエントリーの直後のエントリーから開始します。

リストの順序とアプリケーションでの特定のエントリーの取得方法について詳しくは、34 ページの『List options for QUERY_* Verbs』を参照してください。

dlc_name
情報が必要な DLC の名前、または DLC のリストへの索引として使用される名前。 *list_options* を **リストの最初のリスト** （ _R ）に設定すると、このパラメーターは無視されます。 この名前は 8 バイトの ASCII ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

dlc_summary.overlay_size
The size of the returned `dlc_summary` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `dlc_summary` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

dlc_summary.dlc_name
DLC name. The name is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 characters.

dlc_summary.description
A null-terminated text string describing the DLC, as specified in the definition of the DLC.

dlc_summary.state
State of the DLC. This is one of the following:

AP_ACTIVE
The DLC is active.

AP_NOT_ACTIVE
The DLC is not active.

AP_PENDING_INACTIVE
STOP_DLC is in progress.

dlc_summary.dlc_type
Type of DLC. This is one of the following:

AP_SDLC
SDLC

AP_X25
QLLC

AP_TR
Token Ring

AP_ETHERNET
Ethernet

AP_MPC
Multipath Channel (MPC) adapter, CS Linux for IBM Z only

AP_IP
Enterprise Extender (HPR/IP)

dlc_detail.overlay_size
The size of the returned `dlc_detail` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `dlc_detail` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

dlc_detail.dlc_name
DLC name. The name is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 characters.

dlc_detail.det_data.state
State of the DLC. This is one of the following:

AP_ACTIVE
The DLC is active.

AP_NOT_ACTIVE
The DLC is not active.

AP_PENDING_INACTIVE
STOP_DLC is in progress.

dlc_detail.def_data.description
A null-terminated text string describing the DLC, as specified in the definition of the DLC.

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

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

AP_NO
The DLC is not automatically started; it must be started manually.

dlc_detail.def_data.dlc_type
Type of DLC. This is one of the following:

AP_SDLC
SDLC

AP_X25
QLLC

AP_TR
Token Ring

AP_ETHERNET
Ethernet

AP_IP
Enterprise Extender (HPR/IP)

dlc_detail.def_data.neg_ls_supp
Specifies whether the DLC supports negotiable link stations. Possible values are:

AP_YES
Link stations using this DLC may be negotiable.

AP_NO
Link stations using this DLC must be defined as either primary or secondary; negotiable link stations are not supported.

dlc_detail.def_data.port_types
If *dlc_type* is set to AP_TR / AP_ETHERNET/ AP_IP, this parameter will be set to AP_PORT_SATF. For other DLC types, this parameter is reserved.

dlc_detail.def_data.hpr_only
Specifies whether the DLC is used for Enterprise Extender links and therefore supports only HPR traffic. Possible values are:

AP_YES
This DLC is used for Enterprise Extender links, and supports only HPR traffic.

AP_NO
This DLC is not used for Enterprise Extender links, and supports non-HPR traffic; it may also support HPR traffic.

dlc_detail.def_data.dlc_spec_data_len
Unpadded length, in bytes, of data specific to the type of DLC. The data structure for this data follows the def_data structure, but is padded to start on a 4-byte boundary. For more details of the DLC-specific data, see "DEFINE_DLC" on page 77.

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_DLC_NAME
The *list_options* parameter was set to AP_LIST_INCLUSIVE, to list all entries starting from the supplied name, but the *dlc_name* parameter was not valid.

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

Appendix B, "共通戻りコード," on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, "共通戻りコード," on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

QUERY_DLC_TRACE

QUERY_DLC_TRACE returns information about DLC line tracing, which was set up using ADD_DLC_TRACE verbs.

This verb 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.

VCB structure

```
typedef struct query_dlc_trace
{
    AP_UINT16          opcode;              /* Verb operation code          */
    unsigned char      reserv2;             /* reserved                     */
    unsigned char      format;              /* reserved                     */
    AP_UINT16          primary_rc;          /* Primary return code          */
    AP_UINT32          secondary_rc;        /* Secondary return code        */
    unsigned char      *buf_ptr;            /* pointer to buffer            */
    AP_UINT32          buf_size;            /* buffer size                  */
    AP_UINT32          total_buf_size;      /* total buffer size required   */
    AP_UINT16          num_entries;         /* number of entries            */
    AP_UINT16          total_num_entries;   /* total number of entries      */
    unsigned char      list_options;        /* listing options              */
    unsigned char      list_type;           /* type of listing required     */
    DLC_TRACE_FILTER   filter_entry;        /* resource to start at         */
} QUERY_DLC_TRACE;
```

```
typedef struct dlc_trace_data
{
    AP_UINT16          overlay_size;        /* size of returned entry       */
    DLC_TRACE_FILTER   filter;              /* DLC trace filter information */
} DLC_TRACE_DATA;
```

```
typedef struct dlc_trace_filter
{
    unsigned char      resource_type;       /* type of resource             */
    unsigned char      resource_name[8];    /* name of resource             */
    SNA_LFSID          lfsid;               /* session identifier           */
    unsigned char      message_type;        /* type of messages             */
} DLC_TRACE_FILTER;
```

```
typedef struct sna_lfsid
{
      union
      {
           AP_UINT16            session_id;
           struct
           {
              unsigned char     sidh;
              unsigned char     sidl;
           } s;
      } uu;
      AP_UINT16                 odai;
} SNA_LFSID;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_DLC_TRACE

buf_ptr
A pointer to a data buffer that CS Linux will use to return the requested information.

buf_size
Size of the supplied data buffer.

num_entries
Maximum number of DLC_TRACE entries for which data should be returned. To request data for a specific entry rather than a range, specify the value 1. To return as many entries as possible, specify zero; in this case, CS Linux will return the maximum number of entries that can be accommodated in the supplied data buffer.

list_options
The position in the list of DLC_TRACE entries from which CS Linux should begin to return data. Possible values are:

AP_FIRST_IN_LIST
Start at the first entry in the list.

AP_LIST_INCLUSIVE
Start at the entry specified by the `filter_entry` structure.

AP_LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the `filter_entry` structure.

For more information about how the list is ordered and how the application can obtain specific entries from it, see "List options for QUERY_* Verbs" on page 34.

list_type
The type of resource for which to list tracing options. Possible values are:

AP_ALL_DLC_TRACES
List all specified tracing options (for any resource type).

AP_ALL_RESOURCES
List the tracing options specified for all resources (defined using ADD_DLC_TRACE with a resource type of AP_ALL_RESOURCES).

AP_DLC
List tracing options for DLC resources.

AP_PORT
List tracing options for port resources for which all LSs are traced.

AP_LS
List tracing options for LS resources.

AP_RTP_RESOURCE_TYPE
List tracing options for RTP connection resources.

AP_PORT_DEFINED_LS
List tracing options for port resources for which only defined LSs (not implicit LSs) are traced.

AP_PORT_IMPLICIT_LS
List tracing options for port resources for which only implicit LSs (not defined LSs) are traced.

filter_entry.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 AP_ALL_DLC_TRACES and *list_options* is not set to AP_FIRST_IN_LIST. Possible values are:

AP_ALL_RESOURCES
The required entry specifies the options used for tracing all DLCs, ports, and LSs.

AP_DLC
The required entry specifies tracing options for the DLC named in *resource_name*, and for all ports and LSs that use this DLC.

AP_PORT
The required entry specifies tracing options for the port named in *resource_name*, and for all LSs that use this port.

AP_LS
The required entry specifies tracing options for the LS named in *resource_name*.

AP_RTP_RESOURCE_TYPE
The required entry specifies tracing options for the RTP connection named in the *resource_name* parameter.

AP_PORT_DEFINED_LS
The required entry specifies tracing options for the port named in *resource_name*, and for all defined LSs (but not implicit LSs) that use this port.

AP_PORT_IMPLICIT_LS
The required entry specifies tracing options for the port named in *resource_name*, and for all implicit LSs (but not defined LSs) that use this port.

filter_entry.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 AP_FIRST_IN_LIST, or if *resource_type* is set to AP_ALL_RESOURCES.

filter_entry.lfsid
The Local Form Session Identifier for a session on the specified LS. This is only valid for *resource_type* AP_LS, and indicates that the required entry specifies messages on a particular session for the specified LS. The structure contains the following three values, which are returned in the SESSION_STATS section of a QUERY_SESSION verb:

filter_entry.lfsid.uu.s.sidh
Session ID high byte.

filter_entry.lfsid.uu.s.sidl
Session ID low byte.

filter_entry.lfsid.odai
Origin Destination Assignor Indicator.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer contains the following parameters:

overlay_size
The size of the returned dlc_trace_data structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each dlc_trace_data structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C sizeof() operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

dlc_trace_filter.resource_type
The type of resource being traced. This can take one of the following values:

ALL_RESOURCES
The entry specifies tracing options for all resources.

AP_DLC
The entry specifies tracing options for the DLC named in *resource_name*, and for all ports and LSs that use this DLC.

AP_PORT
The entry specifies tracing options for the port named in *resource_name*, and for all LSs that use this port.

AP_LS
The entry specifies tracing options for the LS named in *resource_name* (or for a particular LFSID on this LS).

AP_RTP_RESOURCE_TYPE
The entry specifies tracing options for the RTP connection named in *resource_name*.

AP_PORT_DEFINED_LS
The entry specifies tracing options for the port named in *resource_name*, and for all defined LSs (but not implicit LSs) that use this port.

AP_PORT_IMPLICIT_LS
The entry specifies tracing options for the port named in *resource_name*, and for all implicit LSs (but not defined LSs) that use this port.

dlc_trace_filter.resource_name
The name of the DLC, port, or LS being traced.

dlc_trace_filter.lfsid
The Local Form Session Identifier for a session on the specified LS. This is only valid for *resource_type* AP_LS, and indicates that only messages on this session are to be traced. The structure contains the following three values, which are returned in the SESSION_STATS section of a QUERY_SESSION verb:

dlc_trace_filter.lfsid.uu.s.sidh
Session ID high byte.

dlc_trace_filter.lfsid.uu.s.sidl
Session ID low byte.

dlc_trace_filter.lfsid.odai
Origin Destination Assignor Indicator.

dlc_trace_filter.message_type
The type of messages being traced for the specified resource or session. This parameter is set to AP_TRACE_ALL to trace all messages, or to one or more of the following values (combined using a logical OR):

AP_TRACE_XID
XID messages

AP_TRACE_SC
Session Control RUs

AP_TRACE_DFC
Data Flow Control RUs

AP_TRACE_FMD
FMD messages

AP_TRACE_NLP
(this message type is currently not used)

AP_TRACE_NC
(this message type is currently not used)

AP_TRACE_SEGS
Non-BBIU segments that do not contain an RH

AP_TRACE_CTL
Messages other then MUs and XIDs

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のいずれかを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

追加のリスト・リスト・タイプ
リスト・タイプ パラメーターに、無効な値が指定されました。

ソース・リソース・タイプのタイプ
リソース・タイプ パラメーターに、無効な値が指定されました。

追加の再リソースが定義されていない
リソース・タイプ パラメーターが **リソースの追加** （ _R ）に設定されていましたが、すべてのリソースのトレース・オプションに対して DLC_TRACE 項目が定義されていません。

追加情報 _RTP_CONNECTION
リソース名 パラメーターで指定された RTP 接続には、トレース・オプションが設定されていません。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

QUERY_DLUR_DEFAULTS

The QUERY_DLUR_DEFAULTS verb allows the user to query the defaults defined using the DEFINE_DLUR_DEFAULTS verb.

VCB structure

```
typedef struct query_dlur_defaults
{
    AP_UINT16         opcode;                  /* verb operation code      */
    unsigned char     reserv2;                 /* reserved                 */
    unsigned char     format;                  /* reserved                 */
    AP_UINT16         primary_rc;              /* primary return code      */
    AP_UINT32         secondary_rc;            /* secondary return code    */
    unsigned char     description[32];         /* resource description     */
    unsigned char     reserv1[16];             /* reserved                 */
    unsigned char     dlus_name[17];           /* DLUS name                */
    unsigned char     bkup_dlus_name[17];      /* Backup DLUS name         */
    unsigned char     reserv3;                 /* reserved                 */
    AP_UINT16         dlus_retry_timeout;      /* DLUS retry timeout       */
    AP_UINT16         dlus_retry_limit;        /* DLUS retry limit         */
    unsigned char     prefer_active_dlus;      /* reserved                  */
    unsigned char     persistent_pipe_support; /* reserved                 */
    unsigned char     reserv4[14];             /* reserved                 */
} QUERY_DLUR_DEFAULTS;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
デフォルトでは AP_QUERY_DLUR_DEFAULTS

記述
リソースの説明。 このパラメーターの長さは、 4 バイトの倍数で、ゼロ以外の値です。

dlus_name
デフォルトの DLUS ノードの名前。 この名前は、すべてゼロまたは 17 バイトの EBCDIC ストリングに設定され、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

bkup_dlus_name
バックアップのデフォルトとして機能する DLUS ノードの名前。 この名前は、すべてゼロまたは 17 バイトの EBCDIC ストリングに設定され、右側には EBCDIC のスペースが埋め込まれます。 これは、

最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

dlus_retry_timeout
2 番目以降の DLUS との接続を試行する間隔 (秒単位) 。 最初の試行と最初の再試行の間隔は常に 1 秒です。

dlus_retry_limit
初期障害後の再試行の最大回数 (DLUS に接続) 。 値 0xFFFF は、 CS Linux が無期限に再試行することを示し

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameter:

primary_rc
AP_OK

戻りパラメーター: 関数はサポートされません

ローカル・ノード構成がそれをサポートしていないために verb が正常に実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc

付加機能がサポートされていません
ローカル・ノードは DLUR をサポートしていません。これは、 DEFINE_NODE verb の *dlur_support* パラメーターによって定義されます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

QUERY_DLUR_LU

QUERY_DLUR_LU returns information about active LUs that are using the DLUR feature of CS Linux. This verb can be used to obtain information about a specific LU, or about multiple LUs, depending on the options used.

This verb must be issued to a running node.

VCB structure

```
typedef struct query_dlur_lu
{
    AP_UINT16         opcode;              /* verb operation code          */
    unsigned char     reserv2;             /* reserved                     */
    unsigned char     format;              /* reserved                     */
    AP_UINT16         primary_rc;          /* primary return code          */
    AP_UINT32         secondary_rc;        /* secondary return code        */
    unsigned char     *buf_ptr;            /* pointer to buffer            */
    AP_UINT32         buf_size;            /* buffer size                  */
    AP_UINT32         total_buf_size;      /* total buffer size required   */
    AP_UINT16         num_entries;         /* number of entries            */
    AP_UINT16         total_num_entries;   /* total number of entries      */
    unsigned char     list_options;        /* listing options              */
    unsigned char     reserv3;             /* reserved                     */
    unsigned char     lu_name[8];          /* LU name                      */
    unsigned char     pu_name[8];          /* PU name filter               */
    unsigned char     filter;              /* local / downstream filter    */
} QUERY_DLUR_LU;
```

```
typedef struct dlur_lu_summary
{
    AP_UINT16         overlay_size;        /* size of returned entry       */
```

```
    unsigned char     lu_name[8];           /* LU name                        */
} DLUR_LU_SUMMARY;
```

```
typedef struct dlur_lu_detail
{
    AP_UINT16         overlay_size;         /* size of returned entry         */
    unsigned char     lu_name[8];           /* LU name                        */
    unsigned char     pu_name[8];           /* PU name of owning PU           */
    unsigned char     dlus_name[17];        /* DLUS name if SSCP-LU session   */
                                            /*  active                        */
    unsigned char     lu_location;          /* downstream or local LU         */
    unsigned char     nau_address;          /* NAU address of LU              */
    unsigned char     plu_name[17];         /* PLU name if PLU-SLU session    */
                                            /*  active                        */
    unsigned char     reserv1[27];          /* reserved                       */
    unsigned char     rscv_len;             /* length of appended RSCV        */
} DLUR_LU_DETAIL;
```

Note : The DLUR_LU_DETAIL structure may be followed by a 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 RSCVs should be stored for DLUR sessions and if the PLU-SLU session is active.

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_DLUR_LU

buf_ptr
A pointer to a data buffer that CS Linux will use to return the requested information.

buf_size
Size of the supplied data buffer.

num_entries
Maximum number of DLUR LUs for which data should be returned. To request data for a specific LU rather than a range, specify the value 1. To return as many entries as possible, specify zero; in this case, CS Linux will return the maximum number of entries that can be accommodated in the supplied data buffer.

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

AP_SUMMARY
Summary information only.

AP_DETAIL
Detailed information.

Combine this value using a logical OR operation with one of the following values:

AP_FIRST_IN_LIST
Start at the first entry in the list.

AP_LIST_INCLUSIVE
Start at the entry specified by the combination of the *pu_name* and *lu_name* parameters.

AP_LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the combination of the *pu_name* and *lu_name* parameters.

The list is ordered by *pu_name* and then by *lu_name*. For more information about how the application can obtain specific entries from the list, see “List options for QUERY_* Verbs” on page 34.

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 AP_FIRST_IN_LIST. The name is an 8-byte EBCDIC type-A string, padded on the right with EBCDIC spaces if the name is shorter than 8 characters.

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, set this field to binary zeros. The name is an 8-byte EBCDIC type-A string, padded on the right with EBCDIC spaces if the name is shorter than 8 characters.

filter
Specifies whether to filter the returned LUs according to their location. Allowed values for network node:

AP_INTERNAL
Return information only for internal LUs.

AP_DOWNSTREAM
Return information only for downstream LUs.

AP_NONE
Return information about all LUs irrespective of location.

For end node, this parameter is reserved (downstream DLUR LUs are not supported).

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

dlur_lu_summary.overlay_size
The size of the returned `dlur_lu_summary` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `dlur_lu_summary` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

dlur_lu_summary.lu_name
Name of the LU. The name is an 8-byte EBCDIC type-A string, padded on the right with EBCDIC spaces if the name is shorter than 8 characters.

dlur_lu_detail.overlay_size
The size of the returned `dlur_lu_detail` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `dur_lu_detail` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

dlur_lu_detail.lu_name
Name of the LU. The name is an 8-byte EBCDIC type-A string, padded on the right with EBCDIC spaces if the name is shorter than 8 characters.

dlur_lu_detail.pu_name
Name of PU associated with the LU. This is an 8-byte type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces.

dlur_lu_detail.dlus_name
If the SSCP-LU session is active, this field contains the name of the DLUS node used by the LU; otherwise it is set to 17 binary zeros. The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

dlur_lu_detail.lu_location
Location of LU.

This is set to one of the following.

AP_INTERNAL
LU is on the local node.

AP_DOWNSTREAM
LU is on a downstream node (network node only).

dlur_lu_detail.nau_address
Network accessible unit address of the LU.

dlur_lu_detail.plu_name
If the PLU-SLU session is active, this field contains the name of the PLU; otherwise it is set to 17 binary zeros. The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

dlur_lu_detail.rscv_len
Length of the RSCV that is appended to the `dlur_lu_detail` structure. If the node's configuration specifies that DLUR RSCVs are not stored, or if the PLU-SLU session is not active, this length is set to zero and no RSCV is included.

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_LU_NAME
The *list_options* parameter was set to AP_LIST_INCLUSIVE to list all entries starting from the supplied name, but the *lu_name* parameter was not valid.

AP_INVALID_FILTER_OPTION
The *filter* parameter was not set to a valid value.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: function not supported

If the verb does not execute successfully because the local node configuration does not support it, CS Linux returns the following parameter:

primary_rc

AP_FUNCTION_NOT_SUPPORTED
The local node does not support DLUR; this is defined by the *dlur_support* parameter on the DEFINE_NODE verb.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

照会 DLURL_PU

QUERY_DLUR_PU は、 CS Linux の DLUR 機能を使用する PU に関する情報を戻します。

この verb は、使用されるオプションに応じて、特定の PU に関する情報、または複数の PU に関する情報を取得するために使用できます。

この verb を非アクティブ・ノードに発行すると、ローカル・ノードで定義された PU に関する情報のみが戻されます。実行中のノードに発行される場合は、このノードで DLUR を使用してアクティブなダウンストリーム PU に関する情報も戻します。

VCB structure

```
typedef struct query_dlur_pu
{
   AP_UINT16        opcode;                 /* verb operation code          */
   unsigned char    reserv2;                /* reserved                     */
   unsigned char    format;                 /* reserved                     */
   AP_UINT16        primary_rc;             /* primary return code          */
   AP_UINT32        secondary_rc;           /* secondary return code        */
   unsigned char    *buf_ptr;               /* pointer to buffer            */
   AP_UINT32        buf_size;               /* buffer size                  */
   AP_UINT32        total_buf_size;         /* total buffer size required   */
   AP_UINT16        num_entries;            /* number of entries            */
   AP_UINT16        total_num_entries;      /* total number of entries      */
   unsigned char    list_options;           /* listing options              */
   unsigned char    reserv3;                /* reserved                     */
   unsigned char    pu_name[8];             /* PU name                      */
   unsigned char    dlus_name[17];          /* fully-qualified DLUS name    */
   unsigned char    filter;                 /* local / downstream filter    */
} QUERY_DLUR_PU;
```

```
typedef struct dlur_pu_summary
{
   AP_UINT16        overlay_size;           /* size of returned entry       */
   unsigned char    pu_name[8];             /* PU name                      */
   unsigned char    description[32];        /* resource description         */
   unsigned char    reserv1[16];            /* reserved                     */
} DLUR_PU_SUMMARY;
```

```
typedef struct dlur_pu_detail
{
   AP_UINT16        overlay_size;           /* size of returned entry       */
   unsigned char    pu_name[8];             /* PU name                      */
   unsigned char    description[32];        /* resource description         */
   unsigned char    initially_active;       /* is the PU initially active?  */
   unsigned char    reserv1[15];            /* reserved                     */
   unsigned char    defined_dlus_name[17];  /* defined DLUS name            */
   unsigned char    bkup_dlus_name[17];     /* backup DLUS name             */
   unsigned char    pu_id[4];               /* PU identifier                */
   unsigned char    pu_location;            /* downstream or local PU       */
   unsigned char    active_dlus_name[17];   /* active DLUS name             */
   unsigned char    ans_support;            /* auto network shutdown support*/
   unsigned char    pu_status;              /* status of the PU             */
```

```
   unsigned char    dlus_session_status;     /* status of the DLUS pipe      */
   unsigned char    reserv3;                 /* reserved                     */
   FQPCID           fqpcid;                  /* FQPCID used on pipe          */
   AP_UINT16        dlus_retry_timeout;      /* DLUR retry timeout           */
   AP_UINT16        dlus_retry_limit;        /* DLUR retry limit             */
} DLUR_PU_DETAIL;
```

```
typedef struct fqpcid
{
   unsigned char    pcid[8];              /* procedure correlator identifier */
   unsigned char    fqcp_name[17];        /* originator's network qualified  */
                                          /*  CP name                        */
   unsigned char    reserve3[3];          /* reserved                        */
} FQPCID;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
照会の追加 （ U ）

buf_ptr
CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

num_entries
データが戻される DLUR PU の最大数。 範囲ではなく、特定の PU のデータを要求するには、 1 という値を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、 CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータの戻りを開始する必要のあるリスト内の位置、および各項目に必要な情報のレベル。 以下の値のいずれかを使用して、情報のレベルを指定します。

要約の要約
サマリー情報のみ。

追加の詳細
詳細情報。

以下のいずれかの値を指定して、論理 **それとも** 操作を使用してこの値を結合します。

リストの最初のリスト （ _R ）
リストの最初の項目から開始します。

リストを含む （ 包括的 ）
プール名 パラメーターで指定されたエントリーから開始します。

次への AP_LIST_FROM_NEXT
プール名 パラメーターによって指定されたエントリーの直後のエントリーから開始します。

リストは プール名によって配列されます。 アプリケーションがリストから特定の項目を取得する方法について詳しくは、 34 ページの『List options for QUERY_* Verbs』を参照してください。

プール名
情報が必要な PU の名前、または PU のリストの索引として使用される名前。 *list_options* を **リストの最初のリスト** （ _R ）に設定すると、この値は無視されます。 この名前は 8 バイトの EBCDIC タイプ A ストリングで、名前が 8 文字より短い場合は、右側に EBCDIC のスペースが埋め込まれます。

dlus_name
PU 情報を必要とする DLUS 名。 特定の DLUS に関連付けられている PU に関する情報のみをリストするには、 DLUS 名を指定します。 PU は、指定された DLUS ノードへの SSCP-PU セッションがある場合にのみリストされます。 すべての DLUSs の完全なリストを取得するには、このフィールドを 2 進ゼロに設定します。

この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

フィルター
戻された PU をその位置に従ってフィルタリングするかどうかを指定します。

ネットワーク・ノードに使用できる値 :

内部
内部 PU の場合にのみ情報を戻します。

ダウンストリーム
ダウンストリーム PU に関する情報のみを戻します。

追加なし
ロケーションに関係なく、すべての PU に関する情報を戻します。

エンド・ノードの場合、このパラメーターは予約されています (ダウンストリーム DLUR PU はサポートされません)。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

dlur_pu_summary.overlay_size
The size of the returned `dlur_pu_summary` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `dlur_pu_summary` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

dlur_pu_summary.pu_name
Name of the PU. The name is an 8-byte EBCDIC type-A string, padded on the right with EBCDIC spaces if the name is shorter than 8 characters.

dlur_pu_summary.description
A null-terminated text string describing the PU, as specified in the definition of the PU. If the PU is an active downstream PU, rather than a defined internal PU, this parameter is reserved.

dlur_pu_detail.overlay_size
The size of the returned `dlur_pu_detail` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `dlur_pu_detail` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

dlur_pu_detail.pu_name
Name of the PU. The name is an 8-byte EBCDIC type-A string, padded on the right with EBCDIC spaces if the name is shorter than 8 characters.

dlur_pu_detail.description
A null-terminated text string describing the PU, as specified in the definition of the PU. If the PU is an active downstream PU, rather than a defined internal PU, this parameter is reserved.

dlur_pu_detail.initially_active
Specifies whether this PU is automatically started when the node is started. For a downstream PU, this parameter is reserved. Possible values for an internal PU are:

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

AP_NO
The PU is not automatically started; it must be started manually.

dlur_pu_detail.defined_dlus_name
Name of DLUS node, defined by either a DEFINE_INTERNAL_PU verb or a DEFINE_LS verb (with *dspu_services* set to AP_DLUR).

The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

dlur_pu_detail.bkup_dlus_name
Name of backup DLUS node, defined by either a DEFINE_INTERNAL_PU verb or a DEFINE_LS verb (with *dspu_services* set to AP_DLUR).

The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

dlur_pu_detail.pu_id
PU identifier, either defined on DEFINE_INTERNAL_PU or obtained in an XID from a downstream PU. This is a 4-byte hexadecimal string, consisting of a block number (3 hexadecimal digits) and a node number (5 hexadecimal digits).

dlur_pu_detail.pu_location
Location of PU.

This is set to one of the following.

AP_INTERNAL
PU is on the local node.

AP_DOWNSTREAM
PU is on a downstream node (network node only).

dlur_pu_detail.active_dlus_name
Name of DLUS node that the PU is currently using. If the SSCP-PU session is not active, this field will be set to all binary zeros.

The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

dlur_pu_detail.ans_support
Auto Network Shutdown support, as sent to DLUR from the DLUS at SSCP-PU activation. It specifies whether link-level contact should be continued if the subarea node initiates an auto network shutdown procedure for the SSCP controlling the PU. Possible values are:

AP_CONT
Continue link-level contact

AP_STOP
Stop link-level contact.

This field is reserved if the SSCP-LU session is inactive.

dlur_pu_detail.pu_status
Status of the PU (as seen by DLUR). Possible values are:

AP_RESET
The PU is in reset state.

AP_PEND_ACTPU
The PU is waiting for an ACTPU from the host.

AP_PEND_ACTPU_RSP
Having forwarded an ACTPU to the PU, DLUR is now waiting for the PU to respond to it.

AP_ACTIVE
The PU is active.

AP_PEND_DACTPU_RSP
Having forwarded a DACTPU to the PU, DLUR is waiting for the PU to respond to it.

AP_PEND_INOP
DLUR is waiting for all necessary events to complete before it deactivates the PU.

dlur_pu_detail.dlus_session_status
Status of the DLUS pipe currently being used by the PU. Possible values are:

AP_PENDING_ACTIVE
The pipe is in the process of being activated.

AP_ACTIVE
The pipe is active.

AP_PENDING_INACTIVE
The pipe is in the process of being deactivated.

AP_INACTIVE
The pipe is not active.

dlur_pu_detail.fqpcid.pcid
Procedure Correlator ID used on the pipe. This is an 8-byte hexadecimal string. If the SSCP-PU session is not active this field will be set to binary zeros.

dlur_pu_detail.fqpcid.fqcp_name
Fully qualified Control Point name used on the pipe. If the SSCP-PU session is not active this field will be set to binary zeros.

The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

The combination of the *pcid* and *fqcp_name* parameters uniquely identify 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.

dlur_pu_detail.dlus_retry_timeout
The interval in seconds between the second and subsequent attempts to contact the DLUS specified by the *def_data.dlus_name* and *def_data.bkup_dlus_name* parameters. The interval between the first and second attempts is always 1 second. If zero is specified, then the defaults specified using the DEFINE_DLUR_DEFAULTS verb are used. .

dlur_pu_detail.dlus_retry_limit
Number of attempts to recontact a DLUS after an initial failure. A value of zero indicates that the value from the DEFINE_DLUR_DEFAULTS verb is used. If 0xFFFF is returned, CS Linux will retry indefinitely.

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_PU_NAME
The *list_options* parameter was set to AP_LIST_INCLUSIVE to list all entries starting from the supplied name, but the *pu_name* parameter was not valid.

AP_INVALID_FILTER_OPTION
The *filter* parameter was not set to a valid value.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: function not supported

If the verb does not execute successfully because the local node configuration does not support it, CS Linux returns the following parameters:

primary_rc

AP_FUNCTION_NOT_SUPPORTED
The local node does not support DLUR; this is defined by the *dlur_support* parameter on the DEFINE_NODE verb.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

照会 DLUS

QUERY_DLUS は、 CS Linux の DLUR 機能に認識されている DLUS ノードに関する情報を戻します。 この verb は、パイプ統計 (SSCP-PU セッション統計および SSCP-LU セッション統計) を戻します。 QUERY_ISR_SESSION verb を使用して PLU-SLU セッション統計を取得することができます。

この verb は、使用されるオプションに応じて、特定の DLUS または複数の DLUSs に関する情報を取得するために使用できます。

非アクティブ・ノードに対してこの verb を発行すると、 DEFINE_INTERNAL_PU または DEFINE_DLUR_DEFAULTS を使用して定義された DLUS ノードに関する情報のみが戻されます。実行中のノードに発行された場合は、アクティブな DLUS ノードに関する情報も戻されます。 この DLUS がアクティブでない限り、 DEFINE_DLUR_DEFAULTS を使用して定義されたバックアップ DLUS に関する情報は戻されません。

VCB structure

```
typedef struct query_dlus
{
    AP_UINT16       opcode;                 /* verb operation code          */
    unsigned char   reserv2;                /* reserved                     */
```

```
   unsigned char   format;                 /* reserved                      */
   AP_UINT16       primary_rc;             /* primary return code           */
   AP_UINT32       secondary_rc;           /* secondary return code         */
   unsigned char   *buf_ptr;               /* pointer to buffer             */
   AP_UINT32       buf_size;               /* buffer size                   */
   AP_UINT32       total_buf_size;         /* total buffer size required    */
   AP_UINT16       num_entries;            /* number of entries             */
   AP_UINT16       total_num_entries;      /* total number of entries       */
   unsigned char   list_options;           /* listing options               */
   unsigned char   reserv3;                /* reserved                      */
   unsigned char   dlus_name[17];          /* fully-qualified DLUS name     */
} QUERY_DLUS;
```

```
typedef struct dlus_data
{
   AP_UINT16       overlay_size;           /* size of returned entry        */
   unsigned char   dlus_name[17];          /* fully qualified DLUS name     */
   unsigned char   is_default;             /* is the DLUS the default       */
   unsigned char   is_backup_default;      /* is DLUS the backup default    */
   unsigned char   pipe_state;             /* state of CPSVRMGR pipe        */
   AP_UINT16       num_active_pus;         /* num of active PUs using pipe  */
   PIPE_STATS      pipe_stats;             /* pipe statistics               */
   unsigned char   persistent_pipe_support; /* reserved                     */
   unsigned char   persistent_pipe;        /* reserved                      */
} DLUS_DATA;
```

```
typedef struct pipe_stats
{
   AP_UINT32       reqactpu_sent;          /* REQACTPUs sent to DLUS        */
   AP_UINT32       reqactpu_rsp_received;  /* RSP(REQACTPU)s received       */
                                           /*  from DLUS                    */
   AP_UINT32       actpu_received;         /* ACTPUs received from DLUS     */
   AP_UINT32       actpu_rsp_sent;         /* RSP(ACTPU)s sent to DLUS      */
   AP_UINT32       reqdactpu_sent;         /* REQDACTPUs sent to DLUS       */
   AP_UINT32       reqdactpu_rsp_received; /* RSP(REQDACTPU)s received      */
                                           /*  from DLUS                    */
   AP_UINT32       dactpu_received;        /* DACTPUs received from DLUS    */
   AP_UINT32       dactpu_rsp_sent;        /* RSP(DACTPU)s sent to DLUS     */
   AP_UINT32       actlu_received;         /* ACTLUs received from DLUS     */
   AP_UINT32       actlu_rsp_sent;         /* RSP(ACTLU)s sent to DLUS      */
   AP_UINT32       dactlu_received;        /* DACTLUs received from DLUS    */
   AP_UINT32       dactlu_rsp_sent;        /* RSP(DACTLU)s sent to DLUS     */
   AP_UINT32       sscp_pu_mus_rcvd;       /* MUs for SSCP-PU sessions rcvd */
   AP_UINT32       sscp_pu_mus_sent;       /* MUs for SSCP-PU sessions sent */
   AP_UINT32       sscp_lu_mus_rcvd;       /* MUs for SSCP-LU sessions rcvd */
   AP_UINT32       sscp_lu_mus_sent;       /* MUs for SSCP-LU sessions sent */
} PIPE_STATS;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
AP_QUERY_DLUS

buf_ptr
CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

num_entries
データが戻される DLUSs の最大数。 範囲ではなく、特定の DLUS のデータを要求するには、 1 という値を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、 CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータの戻りを開始する必要があるリスト内の位置。 次の値のいずれかを指定してください。

リストの最初のリスト （ _R ）
リストの最初の項目から開始します。

リストを含む （ 包括的 ）
dlus_name パラメーターで指定されたエントリーから開始します。

次への AP_LIST_FROM_NEXT
dlus_name パラメーターによって指定されたエントリーの直後のエントリーから開始します。

リストは *dlus_name* によって配列されます。 アプリケーションがリストから特定の項目を取得する方法について詳しくは、 34 ページの『List options for QUERY_* Verbs』を参照してください。

dlus_name
情報が必要な DLUS の名前、または DLUSs のリストへの索引として使用される名前。 *list_options* を **リストの最初のリスト** （ _R ）に設定すると、この値は無視されます。

この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

buf_size
提供されたバッファーに戻された情報の長さ。

total_buf_size
要求されたすべてのリスト情報を戻すために必要となるバッファーのサイズを示す戻り値。 *buf_size* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

num_entries
データ・バッファーに戻されたエントリーの数。

total_num_entries
使用可能な項目の合計数。 *num_entries* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

データ・バッファー内の各項目は、以下のパラメーターで構成されます。

dlus_data.オーバーレイ・サイズ
戻された dlus_data 構造体のサイズ。すなわち、データ・バッファー内の次のエントリーの先頭までのオフセット。

アプリケーションが戻されたバッファーを調べて、各 dlus_data 構造体を順番に検出する必要がある場合は、この値を使用して次のデータ構造の正しいオフセットに移動しなければなりません。また、 C sizeof() 演算子を使用してはなりません。 これは、返されるオーバーレイのサイズが 、 CS Linux の将来のリリースで増加する可能性があるためです。戻されたオーバーレイ・サイズを使用すると、アプリケーションが今後のリリースで引き続き動作することが保証されます。

dlus_data.dlus_name
DLUS の名前。 この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

dlus_data.is_default
DLUS ノードが DEFINE_DLUR_DEFAULTS verb (**類人猿** または **アプ・ ノー**) によってデフォルトとして指定されているかどうかを指定します。

dlus_data.is_backup_default
DLUS ノードが DEFINE_DLUR_DEFAULTS verb (**類人猿** または **アプ・ ノー**) によってバックアップ・デフォルトとして指定されているかどうかを指定します。

dlus_data.pipe_state
パイプの状態 (DLUS) 。 可能な値は次のとおりです

アクティブの追加 （ _L ）
パイプは活動化の進行中です。

アクティブ （ アクティブ ）
パイプはアクティブです。

年金を非アクティブにする
パイプは非活動化の処理中です。

非アクティブ
パイプはアクティブではありません。

dlus_data.num_active_pus
現在、 DLUS にパイプを使用している PU の数。

dlus_data.pipe_stats.reqactpu_sent
パイプ上で DLUS に送信された REQACTPU の数。

dlus_data.pipe_stats.reqactpu_rsp_受信済み
パイプ上で DLUS から受信した RSP(REQACTPU)s の数。

dlus_data.pipe_stats.actpu_受け入れた
パイプ上で DLUS から受信した ACTPU の数。

dlus_data.pipe_stats.actpu_rsp_rsp_sent 」
パイプ上で DLUS に送信された RSP(ACTPU)s の数。

dlus_data.pipepe_stats.reqdactpu_sent
パイプ上で DLUS に送信された REQDACTPU の数。

dlus_data.pipe_stats.reqdactpu_rsp_rsp_受信済み
パイプ上で DLUS から受信した RSP(REQDACTPU)s の数。

dlus_data.pipe_stats.dactpu_受け取った dactpu_受け入れた
パイプ上の DLUS から受信した DACTPU の数。

dlus_data.pipe_stats.dactpu_rsp_rsp_sent 」
パイプ上で DLUS に送信された RSP(DACTPU)s の数。

dlus_data.pipe_stats.actlu_ 受信済み
パイプ上で DLUS から受信した ACTLU の数。

dlus_data.pipe_stats.actlu_rsp_sent
パイプ上で DLUS に送信された RSP(ACTLU)s の数。

dlus_data.pipe_stats.dactlu_譲り受けた
パイプ上で DLUS から受信した DACTLU の数。

***dlus_data.pipe_stats.dactlu_rsp_sent* 送信済み**
パイプ上で DLUS に送信された RSP(DACTLU)s の数。

dlus_data.pipe_stats.sscp_pu_mus_rcvd
パイプ上で DLUS から受信した SSCP-PU CU の数。

dlus_data.pipe_stats.sscp_pu_mus_sent
パイプ上で DLUS に送信された SSCP-PU MU の数。

dlus_data.pipe_stats.sscp_lu_mus_rcvd
パイプ上で DLUS から受信した SSCP-LU MU の数。

dlus_data.pipe_stats.sscp_lu_mus_sent
パイプ上で DLUS に送信された SSCP-LU MU の数。

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_DLUS_NAME
The *list_options* parameter was set to AP_LIST_INCLUSIVE, to list all entries starting from the supplied name, but the *dlus_name* parameter was not valid.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: 関数はサポートされません

ローカル・ノード構成がそれをサポートしていないために verb が正常に実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc

付加機能がサポートされていません
ローカル・ノードは DLUR をサポートしていません。これは、 DEFINE_NODE verb の *dlur_support* パラメーターによって定義されます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

クエリー・ドメイン CONFIG_FILE

QUERY_DOMAIN_CONFIG_FILE は、 CS Linux ドメイン構成ファイル (CS Linux のバージョン番号、ファイルの改訂レベル、および DEFINE_DOMAIN_CONFIG_FILE で提供されるオプションのコメント・ストリング) に含まれるヘッダー情報を戻します。

この verb は、ドメイン構成ファイルに対して発行する必要があります。

VCB structure

```
typedef struct query_domain_config_file
{
    AP_UINT16             opcode;              /* verb operation code       */
    unsigned char         reserv2;             /* reserved                  */
    unsigned char         format;              /* reserved                  */
    AP_UINT16             primary_rc;          /* primary return code       */
    AP_UINT32             secondary_rc;        /* secondary return code     */
    unsigned char         reserv3[8];          /* Reserved                  */
    CONFIG_FILE_HEADER    hdr;
} QUERY_DOMAIN_CONFIG_FILE;
```

```
typedef struct config_file_header
{
    AP_UINT16             major_version;       /* major version number      */
    AP_UINT16             minor_version;       /* minor version number      */
    AP_UINT16             update_release;      /* update release            */
    AP_UINT32             revision_level;      /* file revision number      */
    unsigned char         comment[100];        /* optional comment string   */
    AP_UINT16             updating;            /* reserved                  */
} CONFIG_FILE_HEADER;
```

Supplied parameters

The application supplies the following parameter:

opcode
AP_QUERY_DOMAIN_CONFIG_FILE

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

hdr.major_version*、 *hdr.minor_version*、 *hdr.update_release
このファイルの作成に使用された CS Linux のリリースの内部バージョン ID 。

hdr.revision_level
ファイルの改訂レベル (CS Linux によって内部に保管されます) 。

hdr.comment
ファイルに関する情報を含むオプションのコメント・ストリング。 これは、 0 から 99 文字の ASCII ストリングで、その後にヌル文字が続きます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

QUERY_DOWNSTREAM_LU

QUERY_DOWNSTREAM_LU returns information about downstream LUs that use SNA gateway or DLUR or both. It also returns information about downstream LUs used by applications that communicates with a CS Linux Primary RUI application. For more information about Primary RUI, see IBM Communications Server for Data Center Deployment on AIX or Linux LUA Programmer's Guide.

The returned information is structured as determined data (data gathered dynamically during execution, 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 verb can be used to obtain information about a specific LU, or about multiple LUs, depending on the options used.

VCB structure

```
typedef struct query_downstream_lu
{
   AP_UINT16        opcode;                 /* verb operation code         */
   unsigned char    reserv2;                /* reserved                    */
   unsigned char    format;                 /* reserved                    */
   AP_UINT16        primary_rc;             /* primary return code         */
   AP_UINT32        secondary_rc;           /* secondary return code       */
   unsigned char    *buf_ptr;               /* pointer to buffer           */
   AP_UINT32        buf_size;               /* buffer size                 */
   AP_UINT32        total_buf_size;         /* total buffer size required  */
   AP_UINT16        num_entries;            /* number of entries           */
   AP_UINT16        total_num_entries;      /* total number of entries     */
   unsigned char    list_options;           /* listing options             */
   unsigned char    reserv3;                /* reserved                    */
   unsigned char    dspu_name[8];           /* Downstream PU name filter   */
   unsigned char    dslu_name[8];           /* Downstream LU name          */
   unsigned char    dspu_services;          /* services provided to LU     */
} QUERY_DOWNSTREAM_LU;
```

```
typedef struct downstream_lu_summary
{
   AP_UINT16        overlay_size;           /* size of returned entry      */
   unsigned char    dspu_name[8];           /* PU name                     */
   unsigned char    dslu_name[8];           /* LU name                     */
   unsigned char    description[32];        /* resource description        */
   unsigned char    reserv1[16];            /* reserved                    */
   unsigned char    dspu_services;          /* Type of services provided   */
                                            /*  to downstream LU           */
   unsigned char    nau_address;            /* NAU address                 */
   unsigned char    lu_sscp_sess_active;    /* Is LU-SSCP session active   */
```

```
   unsigned char    plu_sess_active;          /* Is PLU-SLU session active   */
} DOWNSTREAM_LU_SUMMARY;
```

```
typedef struct downstream_lu_detail
{
   AP_UINT16        overlay_size;             /* size of returned entry      */
   unsigned char    dslu_name[8];             /* LU name                     */
   unsigned char    reserv1[2];               /* reserved                    */
   DOWNSTREAM_LU_DET_DATA  det_data;          /* Determined data             */
   DOWNSTREAM_LU_DEF_DATA  def_data;          /* Defined data                */
} DOWNSTREAM_LU_DETAIL;
```

```
typedef struct downstream_lu_det_data
{
   unsigned char    lu_sscp_sess_active;      /* Is LU-SSCP session active   */
   unsigned char    plu_sess_active;          /* Is PLU-SLU session active   */
   unsigned char    dspu_services;            /* Type of service provided to */
                                              /*  downstream node            */
   unsigned char    reserv1;                  /* reserved                    */
   SESSION_STATS    lu_sscp_stats;            /* LU-SSCP session statistics  */
   SESSION_STATS    ds_plu_stats;             /* Downstream PLU-SLU session  */
                                              /*  statistics                 */
   SESSION_STATS    us_plu_stats;             /* Upstream PLU-SLU session    */
                                              /*  statistics                 */
   unsigned char    host_lu_name[8];          /* Determined host LU name     */
   unsigned char    host_pu_name[8];          /* Determined host PU name     */

   unsigned char    reserva[4];               /* reserved                    */
} DOWNSTREAM_LU_DET_DATA;
```

```
typedef struct downstream_lu_def_data
{
   unsigned char    description[32];          /* resource description        */
   unsigned char    reserv1[16];              /* reserved                    */
   unsigned char    nau_address;              /* downstream LU nau address   */
   unsigned char    dspu_name[8];             /* Downstream PU name          */
   unsigned char    host_lu_name[8];          /* Host LU or Pool name        */
   unsigned char    allow_timeout;            /* Allow timeout of host LU    */
   unsigned char    delayed_logon;            /* Allow delayed logon to      */
                                              /*  host LU                    */
   unsigned char    reserv2[6];               /* reserved                    */
} DOWNSTREAM_LU_DEF_DATA;
```

```
typedef struct session_stats
{
  AP_UINT16       rcv_ru_size;          /* session receive RU size           */
  AP_UINT16       send_ru_size;         /* session send RU size              */
  AP_UINT16       max_send_btu_size;    /* maximum send BTU size             */
  AP_UINT16       max_rcv_btu_size;     /* maximum rcv BTU size              */
  AP_UINT16       max_send_pac_win;     /* maximum send pacing window size   */
  AP_UINT16       cur_send_pac_win;     /* current send pacing window size   */
  AP_UINT16       max_rcv_pac_win;      /* maximum receive pacing window size*/
  AP_UINT16       cur_rcv_pac_win;      /* current receive pacing window size*/
  AP_UINT32       send_data_frames;     /* number of data frames sent        */
  AP_UINT32       send_fmd_data_frames; /* num fmd data frames sent          */
  AP_UINT32       send_data_bytes;      /* number of data bytes sent         */
  AP_UINT32       rcv_data_frames;      /* number of data frames received    */
  AP_UINT32       rcv_fmd_data_frames;  /* num fmd data frames received      */
  AP_UINT32       rcv_data_bytes;       /* number of data bytes received     */
  unsigned char   sidh;                 /* session ID high byte (from LFSID) */
  unsigned char   sidl;                 /* session ID low byte (from LFSID)  */
  unsigned char   odai;                 /* ODAI bit set                      */
  unsigned char   ls_name[8];           /* Link station name                 */
  unsigned char   pacing_type;          /* type of pacing in use             */
} SESSION_STATS;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
　下位レベルの追加照会

buf_ptr
　CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

num_entries
データが戻されるダウンストリーム LU の最大数。 特定の範囲ではなく、特定の LU のデータを要求するには、値 1 を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、 CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータの戻りを開始する必要のあるリスト内の位置、および各項目に必要な情報のレベル。 以下の値のいずれかを使用して、情報のレベルを指定します。

要約の要約
サマリー情報のみ。

追加の詳細
詳細情報。

以下のいずれかの値を指定して、論理 **それとも** 操作を使用してこの値を結合します。

リストの最初のリスト （ _R ）
リストの最初の項目から開始します。

リストを含む （ 包括的 ）
dspu_name パラメーターと *dslu_name* パラメーターの組み合わせによって指定されたエントリーから開始します。

次への AP_LIST_FROM_NEXT
dspu_name パラメーターと *dslu_name* パラメーターの組み合わせによって指定されたエントリーの直後のエントリーから開始します。

リストは *dspu_name* によって順序付けられ、次に *dslu_name* によって順序付けられます。 アプリケーションがリストから特定の項目を取得する方法について詳しくは、 34 ページの『List options for QUERY_* Verbs』を参照してください。

dspu_name
LU 情報が必要とされる PU 名 (DEFINE_LS で指定されたもの)。 特定の PU に関連付けられた LU に関する情報のみをリストするには、 PU 名を指定します。 すべての PU の完全なリストを取得するには、このフィールドを 2 進ゼロに設定します。 この名前は 8 バイトの EBCDIC タイプ A ストリングで、名前が 8 文字より短い場合は、右側に EBCDIC のスペースが埋め込まれます。

dslu_name
情報が必要な LU の名前、または LU のリストの索引として使用される名前。 *list_options* を **リストの最初のリスト （ _R ）**に設定すると、この値は無視されます。 この名前は 8 バイトの EBCDIC タイプ A ストリングで、名前が 8 文字より短い場合は、右側に EBCDIC のスペースが埋め込まれます。

dspu_services
DSPU サービス・フィルター。 実行中のノードに対して verb が発行された場合、このパラメーターは、戻される情報を LU に提供されるサービスのタイプ別にフィルターに掛けるかどうかを指定します。 可能な値は次のとおりです

付加プールの集中
SNA ゲートウェイが提供するダウンストリーム LU に関する情報のみを戻します。

アプドラー
DLUR からサービスを受けるダウンストリーム LU のみに関する戻り情報。

追加なし
すべてのダウンストリーム LU に関する情報を返す

ノードが稼働していない場合、このパラメーターは無視され、 CS Linux はすべてのダウンストリーム LU に関する情報を戻します。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

buf_size
提供されたバッファーに戻された情報の長さ。

total_buf_size
要求されたすべてのリスト情報を戻すために必要となるバッファーのサイズを示す戻り値。 *buf_size* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

num_entries
データ・バッファーに戻されたエントリーの数。

total_num_entries
使用可能な項目の合計数。 *num_entries* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

データ・バッファー内の各項目は、以下のパラメーターで構成されます。

lu_summary.オーバーレイ・サイズのダウンストリーム
戻された lu_lu_サマリーのダウン 構造体のサイズ。すなわち、データ・バッファー内の次のエントリーの先頭までのオフセット。

アプリケーションが戻されたバッファーを調べて、各 lu_lu_サマリーのダウン 構造体を順番に検出する必要がある場合は、この値を使用して次のデータ構造の正しいオフセットに移動しなければなりません。また、 C sizeof() 演算子を使用してはなりません。 これは、返されるオーバーレイのサイズが、 CS Linux の将来のリリースで増加する可能性があるためです。戻されたオーバーレイ・サイズを使用すると、アプリケーションが今後のリリースで引き続き動作することが保証されます。

***lu_lu_summary.dspu_name* の要約**
LU に関連付けられている PU の名前。 この名前は 8 バイトの EBCDIC タイプ A ストリングで、名前が 8 文字より短い場合は、右側に EBCDIC のスペースが埋め込まれます。

***lu_lu_summary.dslu_name* の要約**
LU の名前。 この名前は 8 バイトの EBCDIC タイプ A ストリングで、名前が 8 文字より短い場合は、右側に EBCDIC のスペースが埋め込まれます。

***lu_lu_summary.description* の要約**
ダウンストリーム LU の定義に指定されているように、ダウンストリーム LU を記述するヌル終了テキスト・ストリング。

DLUR がサポートしている LU の場合、このパラメーターは予約されています。

***lu_lu_summary.dspu_services* の要約**
実行中のノードに対して verb が発行されると、このパラメーターは、ローカル・ノードによってダウンストリーム LU に提供されるサービスを指定します。

可能な値は次のとおりです

付加プールの集中
ダウンストリーム LU は SNA ゲートウェイによって提供されます。

アプドラー
ダウンストリーム LU は DLUR によってサービスされます。

***lu_summary.nau__* アドレスのダウンストリーム**
LU のネットワーク・アクセス可能な装置アドレス。

***lu_summary.lu_sscp_sess_active* のダウンストリーム**
LU-SSCP セッションがアクティブかどうかを指定します。 可能な値は次のとおりです

類人猿
セッションはアクティブです。

アプ・ノー
セッションはアクティブではありません。

***lu_lu_summary.plu_sess_active* のダウンストリーム**
PLU-SLU セッションがアクティブかどうかを指定します。 可能な値は次のとおりです

類人猿
セッションはアクティブです。

アプ・ノー
セッションはアクティブではありません。

***lu_lu_detail.*オーバーレイ・サイズを削減してください**
戻された lu_detail のダウンストリーム化 構造体のサイズ。すなわち、データ・バッファー内の次のエントリーの先頭までのオフセット。

アプリケーションが戻されたバッファーを調べて、各 lu_detail のダウンストリーム化 構造体を順番に検出する必要がある場合は、この値を使用して次のデータ構造の正しいオフセットに移動しなければなりません。また、 C sizeof() 演算子を使用してはなりません。 これは、返されるオーバーレイのサイズが、 CS Linux の将来のリリースで増加する可能性があるためです。戻されたオーバーレイ・サイズを使用すると、アプリケーションが今後のリリースで引き続き動作することが保証されます。

***lu_lu_detail.dslu_name* をダウンロードしてください。**
LU の名前。 この名前は 8 バイトの EBCDIC タイプ A ストリングで、名前が 8 文字より短い場合は、右側に EBCDIC のスペースが埋め込まれます。

luv_detail.det_data.lu_sscp_sess_active
LU-SSCP セッションがアクティブかどうかを 指定します。 可能な値は次のとおりです

類人猿
セッションはアクティブです。

アプ・ノー
セッションはアクティブではありません。

***lu_lu_detail.det_data.plu_sess_active* のダウンストリーム**
PLU-SLU セッションがアクティブかどうかを 指定します。 可能な値は次のとおりです

類人猿
セッションはアクティブです。

アプ・ノー
セッションはアクティブではありません。

***lu_lu_detail.det_data.dspu_services* のダウンストリーム**
実行中のノードに対して verb が発行されると、このパラメーターは、ローカル・ノードによってダウンストリーム LU に提供されるサービスを指定します。

可能な値は次のとおりです

付加プールの集中
ダウンストリーム LU は SNA ゲートウェイによって提供されます。

アプドラー
ダウンストリーム LU は DLUR によってサービスされます。

3 つのセッション (LU-SSCP セッション、ダウンストリーム PLU-SLU セッション、およびアップストリーム PLU-SLU セッション) のそれぞれについて、 session_stats 構造が組み込まれています。 この構造体のフィールドは以下のとおりです。

rcv_ru_size
最大受信 RU サイズ。 (LU-SSCP セッション統計では、このパラメーターは予約されています。)

send_ru_size
送信 RU の最大サイズ。 (LU-SSCP セッション統計では、このパラメーターは予約されています。)

最大 *send_btu_size*
送信可能な BTU の最大サイズ。

最大 *rcv_btu_size*
受信可能な BTU の最大サイズ。

***send_pac_win* の最大値**
このセッションの「送信ペーシング」ウィンドウの最大サイズ。(LU-SSCP セッション統計では、このパラメーターは予約されています。)

cur_send_pac_win
このセッションの「送信ペーシング」ウィンドウの現在のサイズ。(LU-SSCP セッション統計では、このパラメーターは予約されています。)

最大値の *pac_win*
このセッションの受信ペーシング・ウィンドウの最大サイズ。(LU-SSCP セッション統計では、このパラメーターは予約されています。)

cur_rcv_pac_win
このセッションでの受信ペーシング・ウィンドウの現在のサイズ。(LU-SSCP セッション統計では、このパラメーターは予約されています。)

send_data_frames
送信された通常フロー・データ・フレームの数。

***send_fmd_data_*フレーム**
送信された通常フロー FMD データ・フレームの数。

send_data_bytes
送信された通常フロー・データ・バイトの数。

rcv_data_frames
受信された通常フロー・データ・フレームの数。

***rcv_fmd_data_*フレーム**
受信された通常フロー FMD データ・フレームの数。

rcv_data_bytes
受信された通常フロー・データ・バイトの数。

シダ
セッション ID の高位バイト。(SNA ゲートウェイによって提供される LU についてのアップストリームの PLU-SLU セッション統計では、このパラメーターは予約済みです。)

シドル
セッション ID の低位バイト。(SNA ゲートウェイによって提供される LU についてのアップストリームの PLU-SLU セッション統計では、このパラメーターは予約済みです。)

オーダイ
出荷元宛先の割り当てインディケーター。 セッションを起動すると、ローカル・ノードに 1 次リンク・ステーションが含まれている場合は、 BIND の送信側はこのフィールドをゼロに設定し、 BIND 送信側が 2 次リンク・ステーションを含むノードである場合はそれを 1 つに設定します。(SNA ゲートウェイによって提供される LU についてのアップストリームの PLU-SLU セッション統計では、このパラメーターは予約済みです。)

ls_name
統計に関連したリンク・ステーション名。 これは 8 バイトの ASCII 文字ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。(SNA ゲートウェイによって提供される LU についてのアップストリームの PLU-SLU セッション統計では、このパラメーターは予約済みです。)

データ・タイプの埋め込み
このセッションで使用中の受信ペーシングのタイプ。 可能な値は次のとおりです

- **追加なし**
- **固定の付加 （ _L ）**

***lu_lu_detail.det_data.host_lu_name* がダウンロードされました。**
ダウンストリーム LU がマップされているホスト LU の名前、または PLU-SLU セッションが最後にアクティブであったときにマップされたホスト LU の名前。 データ *.host_lu_name* の更新 は、ホスト LU プールの名前にすることができるため、このパラメーター値は データ *.host_lu_name* の更新 と異なる場合があります。

ダウンストリーム LU がホストの代わりに CS Linux 1 次 RUI アプリケーションと 通信するために使用される場合、このフィールドは EBCDIC でストリング *#*PRIRUI*#* に設定されます。

***lu_lu_detail.det_data.host_pu_name* をダウンロードしてください。**
ダウンストリーム LU がマップされているホスト PU の名前、または PLU-SLU セッションが最後にアクティブであったときにマップされたホスト PU の名前。

***lu_lu_detail.def_data.description* をダウンロードしてください。**
ダウンストリーム LU の定義に指定されているように、ダウンストリーム LU を記述するヌル終了テキスト・ストリング。 DLUR がサポートしている LU の場合、このパラメーターは予約されています。

***lu_lu_detail.def_data.nau_address* をダウンロードしてください**
ダウンストリーム LU のネットワーク・アクセス可能な装置アドレス。

***lu_lu_detail.def_data.dspu_name* のダウンロードが停止中**
この LU に関連付けられているダウンストリーム PU の名前 (DEFINE_LS verb で指定された名前) 。これは 8 バイトのタイプ A の EBCDIC ストリング (文字で始まる) で、名前が 8 文字より短い場合は、右側に EBCDIC のスペースが埋め込まれます。

***lu_detail.def_data.host_lu_name* がダウンロードされました。**
ダウンストリーム LU が使用するホスト LU またはホスト LU プールの名前。 これは 8 バイトの EBCDIC ストリングで、名前が 8 文字より短い場合は、右側に EBCDIC のスペースが埋め込まれます。

ダウンストリーム LU がホストの代わりに CS Linux 1 次 RUI アプリケーションと通信するために使用される場合、このフィールドは EBCDIC でストリング #PRIRUI# に設定されます。

このフィールドは、 DLUR が処理するダウンストリーム LU に予約されます。

***lu_lu_detail.allow_timeout* をダウンロードしてください。**
このダウンストリーム LU が、アップストリーム LU とのセッションをタイムアウトにするかどうかを指定します。 可能な値は次のとおりです

類人猿
このダウンストリーム LU では、アップストリーム LU とのセッションがタイムアウトになります。

アプ・ノー
このダウンストリーム LU は、アップストリーム LU とのセッションをタイムアウトにすることはできません。

ダウンストリーム LU がホストの代わりに CS Linux 1 次 RUI アプリケーションと通信するために使用される場合、このフィールドは無視されます。

***lu_detail.delayed_logon* をダウンロードしてください。**
このダウンストリーム LU が遅延ログオンを使用するかどうかを指定します (アップストリーム LU は、ユーザーが活動化されるまで活動化されません) 。 可能な値は次のとおりです

類人猿
このダウンストリーム LU は遅延ログオンを使用する

アプ・ノー
このダウンストリーム LU は遅延ログオンを使用しません。

ダウンストリーム LU がホストの代わりに CS Linux 1 次 RUI アプリケーションと通信するために使用される場合、このフィールドは無視されます。

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

ファイル名の変更
list_options パラメーターが **リストを含む** (**包括的**) に設定されました。指定された名前から始まるすべての項目をリストしますが、 *lu_name* パラメーターが無効でした。

ファイルの追加リスト・オプション
list_options パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: state check

If the verb does not execute because of a state error, CS Linux returns the following parameters.

primary_rc
AP_STATE_CHECK

secondary_rc

AP_INVALID_PU_TYPE
The PU specified by the *dspu_name* parameter is not a downstream PU.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_STATE_CHECK, which are common to all NOF verbs.

戻りパラメーター: 関数はサポートされません

ローカル・ノード構成がそれをサポートしていないために verb が正常に実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc

付加機能がサポートされていません
ローカル・ノードは SNA ゲートウェイまたは DLUR をサポートしていません。これは、DEFINE_NODE verb の *pu_conc__* サポートパラメーターおよび *dlur_support* パラメーターによって定義されます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

照会のダウンストリーム

QUERY_DOWNSTREAM_PU は、 SNA ゲートウェイまたは DLUR 、あるいはその両方を使用するダウンストリーム PU に関する情報を戻します。 この verb を使用して、使用するオプションに応じて、特定の PU または複数の PU に関する情報を取得することができます。

この verb は実行中のノードに対して発行する必要があります

VCB structure

```
typedef struct query_downstream_pu
{
  AP_UINT16        opcode;              /* verb operation code             */
  unsigned char    reserv2;             /* reserved                        */
  unsigned char    format;              /* reserved                        */
  AP_UINT16        primary_rc;          /* primary return code             */
  AP_UINT32        secondary_rc;        /* secondary return code           */
  unsigned char    *buf_ptr;            /* pointer to buffer               */
  AP_UINT32        buf_size;            /* buffer size                     */
  AP_UINT32        total_buf_size;      /* total buffer size required      */
  AP_UINT16        num_entries;         /* number of entries               */
  AP_UINT16        total_num_entries;   /* total number of entries         */
  unsigned char    list_options;        /* listing options                 */
  unsigned char    reserv3;             /* reserved                        */
  unsigned char    dspu_name[8];        /* Downstream PU name filter       */
  unsigned char    dspu_services;       /* services provided to PU         */
} QUERY_DOWNSTREAM_PU;
```

```
typedef struct downstream_pu_data
{
  AP_UINT16        overlay_size;         /* size of returned entry           */
  unsigned char    dspu_name[8];         /* PU name                          */
  unsigned char    description[32];      /* resource description             */
  unsigned char    reserv1[16];          /* reserved                         */
  unsigned char    ls_name[8];           /* Link name                        */
  unsigned char    pu_sscp_sess_active;  /* Is the PU-SSCP session active    */
  unsigned char    dspu_services;        /* DSPU service type                */
  SESSION_STATS    pu_sscp_stats;        /* SSCP-PU session statistics       */
  unsigned char    reserva[20];          /* reserved                         */
} DOWNSTREAM_PU_DATA;
```

```
typedef struct session_stats
{
  AP_UINT16        rcv_ru_size;          /* session receive RU size          */
  AP_UINT16        send_ru_size;         /* session send RU size             */
  AP_UINT16        max_send_btu_size;    /* maximum send BTU size            */
  AP_UINT16        max_rcv_btu_size;     /* maximum rcv BTU size             */
  AP_UINT16        max_send_pac_win;     /* maximum send pacing window size  */
  AP_UINT16        cur_send_pac_win;     /* current send pacing window size  */
  AP_UINT16        max_rcv_pac_win;      /* maximum receive pacing window    */
                                         /*  size                            */
  AP_UINT16        cur_rcv_pac_win;      /* current receive pacing window    */
                                         /*  size                            */
  AP_UINT32        send_data_frames;     /* number of data frames sent       */
  AP_UINT32        send_fmd_data_frames; /* num fmd data frames sent         */
  AP_UINT32        send_data_bytes;      /* number of data bytes sent        */
  AP_UINT32        rcv_data_frames;      /* number of data frames received   */
  AP_UINT32        rcv_fmd_data_frames;  /* num fmd data frames received     */
  AP_UINT32        rcv_data_bytes;       /* number of data bytes received    */
  unsigned char    sidh;                 /* session ID high byte (from LFSID)*/
  unsigned char    sidl;                 /* session ID low byte (from LFSID) */
  unsigned char    odai;                 /* ODAI bit set                     */
  unsigned char    ls_name[8];           /* Link station name                */
  unsigned char    pacing_type;          /* type of pacing in use            */
} SESSION_STATS;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
ユーザーの追加を停止する

buf_ptr
CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

num_entries
データが戻されるダウンストリーム PU の最大数。 範囲ではなく、特定の PU のデータを要求するには、 1 という値を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、 CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータの戻りを開始する必要があるリスト内の位置。 可能な値は次のとおりです

リストの最初のリスト （ _R ）
リストの最初の項目から開始します。

リストを含む （ 包括的 ）
dspu_name パラメーターで指定されたエントリーから開始します。

次への AP_LIST_FROM_NEXT
dspu_name パラメーターによって指定されたエントリーの直後のエントリーから開始します。

リストの順序とアプリケーションでの特定のエントリーの取得方法について詳しくは、 34 ページの『List options for QUERY_* Verbs』を参照してください。

dspu_name
情報が必要な PU の名前 (DEFINE_LS で指定される) 、または PU のリストの索引として使用する名前。 *list_options* を **リストの最初のリスト** （ _R ）に設定すると、この値は無視されます。 この名前は 8 バイトの EBCDIC タイプ A ストリングで、名前が 8 文字より短い場合は、右側に EBCDIC のスペースが埋め込まれます。

dspu_services
DSPU サービス・フィルター。 PU に提供されるサービスのタイプによって戻される情報をフィルタリングするかどうかを指定します。 可能な値は次のとおりです

付加プールの集中
SNA ゲートウェイによって提供されるダウンストリーム PU に関する情報のみを戻します。

アプドラー
DLUR からサービスを受けるダウンストリーム PU に関する情報のみを戻します。

追加なし
すべてのダウンストリーム PU に関する情報を戻します。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

downstream_pu_data.overlay_size
The size of the returned `downstream_pu_data` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `downstream_pu_data` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

downstream_pu_data.dspu_name
Name of the downstream PU. The name is an 8-byte EBCDIC type-A string, padded on the right with EBCDIC spaces if the name is shorter than 8 characters.

downstream_pu_data.description
A null-terminated text string describing the LS to the downstream PU, as specified in the definition of the LS.

downstream_pu_data.ls_name
Name of the LS used to access the downstream PU. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 characters.

downstream_pu_data.pu_sscp_sess_active
Specifies whether the PU-SSCP session to the downstream PU is active. Possible values are:

AP_YES
The session is active.

AP_NO
The session is not active.

downstream_pu_data.dspu_services
Specifies the type of services provided to the PU.

Possible values are:

AP_PU_CONCENTRATION
Downstream PU is served by SNA gateway.

AP_DLUR
Downstream PU is served by DLUR.

downstream_pu_data.pu_sscp_stats.rcv_ru_size
Maximum receive RU size; this field is reserved (and set to zero) if the downstream PU is served by SNA gateway.

downstream_pu_data.pu_sscp_stats.send_ru_size
Maximum send RU size; this field is reserved (and set to zero) if the downstream PU is served by SNA gateway.

downstream_pu_data.pu_sscp_stats.max_send_btu_size
Maximum BTU size that can be sent.

downstream_pu_data.pu_sscp_stats.max_rcv_btu_size
Maximum BTU size that can be received.

downstream_pu_data.pu_sscp_stats.max_send_pac_win
Reserved (always set to zero).

downstream_pu_data.pu_sscp_stats.cur_send_pac_win
Reserved (always set to zero).

downstream_pu_data.pu_sscp_stats.max_rcv_pac_win
Reserved (always set to zero).

downstream_pu_data.pu_sscp_stats.cur_rcv_pac_win
Reserved (always set to zero).

downstream_pu_data.pu_sscp_stats.send_data_frames
Number of normal flow data frames sent.

downstream_pu_data.pu_sscp_stats.send_fmd_data_frames
Number of normal flow FMD data frames sent.

downstream_pu_data.pu_sscp_stats.send_data_bytes
Number of normal flow data bytes sent.

downstream_pu_data.pu_sscp_stats.rcv_data_frames
Number of normal flow data frames received.

downstream_pu_data.pu_sscp_stats.rcv_fmd_data_frames
Number of normal flow FMD data frames received.

downstream_pu_data.pu_sscp_stats.rcv_data_bytes
Number of normal flow data bytes received.

downstream_pu_data.pu_sscp_stats.sidh
Session ID high byte.

downstream_pu_data.pu_sscp_stats.sidl
Session ID low byte.

downstream_pu_data.pu_sscp_stats.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.

downstream_pu_data.pu_sscp_stats.ls_name
Link station name associated with statistics. This is an 8-byte ASCII character string, right-padded with spaces if the name is shorter than 8 characters.

downstream_pu_data.pacing_type
The type of receive pacing in use on the PU-SSCP. This parameter will always be set to AP_NONE.

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_PU_NAME
The *list_options* parameter was set to AP_LIST_INCLUSIVE, to list all entries starting from the supplied name, but the *dspu_name* parameter was not valid.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: function not supported

If the verb does not execute successfully because the local node configuration does not support it, CS Linux returns the following parameters:

primary_rc

AP_FUNCTION_NOT_SUPPORTED
The local node does not support SNA gateway or DLUR; this is defined by the *pu_conc_support* and *dlur_support* parameters on the DEFINE_NODE verb.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

QUERY_DSPU_TEMPLATE

The QUERY_DSPU_TEMPLATE verb returns information about defined downstream PU templates used for SNA gateway over implicit links.

This verb can be used to obtain information about a specific downstream PU template, or about a number of downstream PU templates, depending on the options used. To obtain information about a specific downstream PU template or multiple downstream PU templates, set the *template_name* parameter. The *template_name* parameter is ignored if the *list_options* parameter is set to AP_FIRST_IN_LIST.

VCB structure

```
typedef struct query_dspu_template
{
   AP_UINT16       opcode;           /* verb operation code               */
   unsigned char   reserv1;          /* reserved                          */
   unsigned char   format;           /* reserved                          */
   AP_UINT16       primary_rc;       /* primary return code               */
   AP_UINT32       secondary_rc;     /* secondary return code             */
   unsigned char   *buf_ptr;         /* pointer to buffer                 */
   AP_UINT32       buf_size;         /* buffer size                       */
   AP_UINT32       total_buf_size;   /* total buffer size required        */
   AP_UINT16       num_entries;      /* number of entries                 */
```

```
   AP_UINT16       total_num_entries; /* total number of entries            */
   unsigned char   list_options;      /* listing options                    */
   unsigned char   reserv3;           /* reserved                           */
   unsigned char   template_name[8];  /* name of DSPU template              */
} QUERY_DSPU_TEMPLATE;
```

```
typedef struct dspu_template_data
{
   AP_UINT16       overlay_size;      /* size of returned entry             */
   unsigned char   template_name[8];  /* name of DSPU template              */
   unsigned char   description[32];   /* resource description               */
   unsigned char   reserv2[16];       /* reserved                           */
   unsigned char   reserv1[12];       /* reserved                           */
   AP_UINT16       max_instance;      /* max active template instance       */
   AP_UINT16       active_instance;   /* current active instances           */
   unsigned char   num_of_dslu_templates;  /* number of DSLU templates      */
} DSPU_TEMPLATE_DATA;
```

Each dspu_template_data structure is followed by one or more downstream LU templates; the number of the downstream LU templates is specified by the *number_of_dslu_templates* parameter. Each downstream LU template has the following format:

```
typedef struct dslu_template_data
{
   AP_UINT16       overlay_size;      /* size of this entry                 */
   unsigned char   reserv1[2];        /* reserved                           */
   DSLU_TEMPLATE   dslu_template;     /* downstream LU template             */
} DSLU_TEMPLATE_DATA;
```

```
typedef struct dslu_template
{
   unsigned char   min_nau;           /* minimum NAU address in range       */
   unsigned char   max_nau;           /* maximum NAU address in range       */
   unsigned char   allow_timeout;     /* allow timeout of host LU?          */
   unsigned char   delayed_logon;     /* allow delayed logon to host LU     */
   unsigned char   reserv1[8];        /* reserved                           */
   unsigned char   host_lu[8];        /* host LU or pool name               */
} DSLU_TEMPLATE;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
テンプレートの追加テンプレート

buf_ptr
CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

num_entries
データが戻されるテンプレートの最大数。 特定の範囲ではなく、特定のテンプレートのデータを要求するには、 1 という値を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、 CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータの戻りを開始する必要があるリスト内の位置。 可能な値は次のとおりです

リストの最初のリスト （ _R ）
リストの最初の項目から開始します。

リストを含む （ 包括的 ）
テンプレート名 パラメーターで指定されたエントリーから開始します。

次への AP_LIST_FROM_NEXT
テンプレート名 パラメーターによって指定されたエントリーの直後のエントリーから開始します。

リストの順序とアプリケーションでの特定のエントリーの取得方法について詳しくは、 34 ページの『List options for QUERY_* Verbs』を参照してください。

テンプレート名
情報が必要な DSPU テンプレートの名前、またはリスト内の索引として使用される名前。 これは、ローカル表示可能文字セットの 8 バイトのストリングです。 *list_options* を **リストの最初のリスト** (_R) に設定すると、このパラメーターは無視されます。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

buf_size
バッファーに戻された情報の長さ。

total_buf_size
要求されたすべてのリスト情報を戻すために必要となるバッファーのサイズを示す戻り値。 これは、*buf_size* より大きい場合があります

num_entries
実際に戻されたエントリーの数。

total_num_entries
戻された可能性がある項目の合計数。 これは、 *num_entries* より大きい場合があります

dspu_template_data.オーバーレイ ・サイズ
このエントリー内のバイト数 (ダウンストリーム LU テンプレートを含む) 、および戻される次のエントリー (存在する場合) までのオフセット。

アプリケーションが戻されたバッファーを調べて、各 dspu_template_data 構造体を順番に検出する必要がある場合は、この値を使用して次のデータ構造の正しいオフセットに移動しなければなりません。また、 C sizeof() 演算子を使用してはなりません。 これは、返されるオーバーレイのサイズが、 CS Linux の将来のリリースで増加する可能性があるためです。戻されたオーバーレイ ・サイズを使用すると、アプリケーションが今後のリリースで引き続き動作することが保証されます。

dspu_template_data.template_name
DSPU テンプレートの名前。

dspu_template_data.description
DEFINE_DSPU_TEMPLATE verb で定義されているリソース記述。

dspu_template_data.max_instance
同時にアクティブにすることができるテンプレートのインスタンスの 最大数。

dspu_template_data.active_instance
現在アクティブになっているテンプレートのインスタンスの 数。

dspu_template_data.num_of_dslu_テンプレート
このダウンストリーム PU テンプレートのダウンストリーム LU テンプレートの数。 このパラメーターの後には、 DSLU テンプレートごとに 1 つずつ *d_of_dslu_* テンプレートの数 項目があります。

dslu_template_data.オーバーレイ ・サイズ
この項目内のバイト数、および戻される次の項目 (ある場合) へのオフセット。

アプリケーションが戻されたバッファーを調べて、各 dslu_template_data 構造体を順番に検出する必要がある場合は、この値を使用して次のデータ構造の正しいオフセットに移動しなければなりません。また、 C sizeof() 演算子を使用してはなりません。 これは、返されるオーバーレイのサイズが、 CS Linux の将来のリリースで増加する可能性があるためです。戻されたオーバーレイ ・サイズを使用すると、アプリケーションが今後のリリースで引き続き動作することが保証されます。

dslu_template_data.min_nau
DSLU テンプレートの範囲内の最小 NAU アドレス。

dslu_template_data.max_nau
DSLU テンプレートの範囲内の最大 NAU アドレス。

dslu_template_data.allow_timeout
セッションがホスト LU 定義で指定されたタイムアウト期間に非アクティブのままである場合に、このダウンストリーム LU が使用するタイムアウト・ホスト LU に CS Linux が許可されるかどうかを示します。 可能な値は次のとおりです

類人猿
CS Linux は、このダウンストリーム LU が使用するホスト LU のタイムアウトを許可されます。

アプ・ノー
CS Linux は、このダウンストリーム LU が使用するホスト LU をタイムアウトにすることはできません

ダウンストリーム LU が、ホストの代わりに CS Linux 1 次 RUI アプリケーションと通信するために使用される場合、このフィールドは無視されます。

dslu_template_data.delayed_logon
ダウンストリーム LU から最初のデータが受信されるまで、ダウンストリーム LU の接続をホスト LU に接続する CS Linux 遅延の有無を示します。 代わりに、シミュレートされたログオン画面がダウンストリーム LU に送信されます。 可能な値は次のとおりです

類人猿
ダウンストリーム LU をホスト LU に接続する CS Linux の遅延。

アプ・ノー
CS Linux は、ダウンストリーム LU のホスト LU への接続を遅らせません。

ダウンストリーム LU が、ホストの代わりに CS Linux 1 次 RUI アプリケーションと通信するために使用される場合、このフィールドは無視されます。

dslu_template_data.host_lu_name
範囲内のすべてのダウンストリーム LU がマップされるホスト LU またはホスト LU プールの名前。

ダウンストリーム LU がホストの代わりに CS Linux 1 次 RUI アプリケーションと通信するために使用される場合、このフィールドは EBCDIC でストリング #PRIRUI# に設定されます。

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc

アプリケーション・テンプレート名の変更
テンプレート名 パラメーターに指定されたテンプレートが無効でした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

QUERY_FOCAL_POINT

QUERY_FOCAL_POINT returns information about the focal point for a specific Management Services category, or about multiple focal points, depending on the options used.

VCB structure

```
typedef struct query_focal_point
{
   AP_UINT16       opcode;                 /* verb operation code            */
   unsigned char   reserv2;                /* reserved                       */
```

```
   unsigned char   format;                 /* reserved                        */
   AP_UINT16       primary_rc;             /* primary return code             */
   AP_UINT32       secondary_rc;           /* secondary return code           */
   unsigned char   *buf_ptr;               /* pointer to buffer               */
   AP_UINT32       buf_size;               /* buffer size                     */
   AP_UINT32       total_buf_size;         /* total buffer size required      */
   AP_UINT16       num_entries;            /* number of entries               */
   AP_UINT16       total_num_entries;      /* total number of entries         */
   unsigned char   list_options;           /* listing options                 */
   unsigned char   reserv3;                /* reserved                        */
   unsigned char   ms_category[8];         /* name of MS category             */
} QUERY_FOCAL_POINT;
```

```
typedef struct fp_data
{
   AP_UINT16       overlay_size;           /* size of returned entry          */
   unsigned char   ms_appl_name[8];        /* focal point application name    */
   unsigned char   ms_category[8];         /* focal point category            */
   unsigned char   description[32];        /* resource description            */
   unsigned char   reserv1[16];            /* reserved                        */
   unsigned char   fp_fqcp_name[17];       /* focal point fully qualified     */
                                           /*  cp name                        */
   unsigned char   bkup_appl_name[8];      /* backup focal point              */
                                           /*  application name               */
   unsigned char   bkup_fp_fqcp_name[17];  /* backup fp fully qualified cp    */
                                           /*  name                           */
   unsigned char   implicit_appl_name[8];  /* implicit focal point appl name  */
   unsigned char   implicit_fp_fqcp_name[17]; /* implicit fp fully qualified  */
                                              /*  cp name                     */
   unsigned char   fp_type;                /* focal point type                */
   unsigned char   fp_status;              /* focal point status              */
   unsigned char   fp_routing;             /* type of MDS routing to use      */
   unsigned char   reserva[20];            /* reserved                        */
   AP_UINT16       number_of_appls;        /* number of applications          */
} FP_DATA;
```

Each `fp_data` structure is followed by one or more application names; the number of these is specified by the *number_of_appls* parameter. Each application name has the following format:

```
   unsigned char   appl_name[8];           /* application name                */
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_FOCAL_POINT

buf_ptr
A pointer to a data buffer that CS Linux will use to return the requested information.

buf_size
Size of the supplied data buffer.

num_entries
Maximum number of entries for which data should be returned. To request data for a specific entry rather than a range, specify the value 1. To return as many entries as possible, specify zero; in this case, CS Linux will return the maximum number of entries that can be accommodated in the supplied data buffer.

list_options
The position in the list of focal points from which CS Linux should begin to return data. Possible values are:

AP_FIRST_IN_LIST
Start at the first entry in the list.

AP_LIST_INCLUSIVE
Start at the entry specified by the *ms_category* parameter.

AP_LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *ms_category* parameter.

For more information about how the list is ordered and how the application can obtain specific entries from it, see “List options for QUERY_* Verbs” on page 34.

ms_category
Management Services category. This parameter is not used if *list_options* is set to AP_FIRST_IN_LIST.

This may be either one of the category names specified in the MS Discipline-Specific Application Programs table of *Systems Network Architecture: Management Services Reference* (see the Bibliography), padded with EBCDIC spaces (0x40), or a user-defined category. A user-defined category name is an 8-byte type-1134 EBCDIC string, padded with EBCDIC spaces (0x40) if necessary.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

buf_size
提供されたバッファーに戻された情報の長さ。

total_buf_size
要求されたすべてのリスト情報を戻すために必要となるバッファーのサイズを示す戻り値。 *buf_size* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

num_entries
データ・バッファーに戻されたエントリーの数。

total_num_entries
使用可能な項目の合計数。 *num_entries* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

データ・バッファー内の各項目は、以下のパラメーターで構成されます。

fp_data.オーバーレイ・サイズ
戻された fp_data 構造体のサイズ。すなわち、データ・バッファー内の次のエントリーの先頭までのオフセット。

アプリケーションが戻されたバッファーを調べて、各 fp_data 構造体を順番に検出する必要がある場合は、この値を使用して次のデータ構造の正しいオフセットに移動しなければなりません。また、 C sizeof() 演算子を使用してはなりません。 これは、返されるオーバーレイのサイズが、 CS Linux の将来のリリースで増加する可能性があるためです。戻されたオーバーレイ・サイズを使用すると、アプリケーションが今後のリリースで引き続き動作することが保証されます。

fp_data.ms_appl_name
現在アクティブなフォーカル・ポイント・アプリケーションの名前。 これは、 システム・ネットワーク体系*:* 管理サービス参照 に指定されている MS 規律特定アプリケーション・プログラム (Bibliography を参照) 、またはタイプ 1134 文字を使用する EBCDIC ストリングのいずれかです。名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

fp_data.ms_カテゴリー
「管理サービス」カテゴリー。 これは、 システム・ネットワーク体系*:* 管理サービス参照 に指定されたカテゴリー名の 1 つ (Bibliography を参照) 、またはタイプ 1134 文字を使用して EBCDIC ストリング(名前が 8 文字より短い場合はスペースを右側に埋め込む) のいずれかです。

fp_data.description
フォーカル・ポイントの定義に指定されているフォーカル・ポイントを記述するヌル終了のテキスト・ストリング。

fp_data.fp_fqcp_name
現在アクティブなフォーカル・ポイントの制御点の完全修飾名。 この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、 1 から 8 文字の A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および 1 から 8 文字の A ストリング文字のネットワーク名で構成されます。

fp_data.bkup_appl_name
バックアップ・フォーカル・ポイント・アプリケーション名。 これは、 システム・ネットワーク体系*:* 管理サービス参照 に指定されている MS 規律特定アプリケーション・プログラム (Bibliography を参照) 、またはタイプ 1134 文字を使用する EBCDIC ストリングのいずれかです。名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

fp_data.bkup_fp_fqcp_name
バックアップ・フォーカル・ポイントの制御点の完全修飾名。 この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、 1 から 8 文字の A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および 1 から 8 文字の A ストリング文字のネットワーク名で構成されます。

fp_data.implicit_appl_name
暗黙フォーカル・ポイント・アプリケーションの名前 (DEFINE_FOCAL_POINT を使用して指定されます) 。 これは、 システム・ネットワーク体系*:* 管理サービス参照 に指定されている MS 規律特定アプリケーション・プログラム (Bibliography を参照) 、またはタイプ 1134 文字を使用する EBCDIC ストリングのいずれかです。名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

fp_data.implicit_fp_fqcp_name
暗黙フォーカル・ポイント (DEFINE_FOCAL_POINT を使用して指定される) の完全修飾制御点名。 この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、 1 から 8 文字の A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および 1 から 8 文字の A ストリング文字のネットワーク名で構成されます。

fp_data.fp_type
フォーカル・ポイントのタイプ。 詳細については、「 IBM システム・ネットワーク体系*:* 管理サービス参照 」 (「参考文献」を参照) を参照してください。 これは、以下のいずれかです。

検討中のプライマリー FP

AP_IMPLICIT_PRIMARY_FP

追加バックアップ _FP

デフォルトのプライマリー· オプション

上位ドメイン （ FP ）

ホスト· ホスト FP

ファイルの追加 （ _P ）

fp_data.fp_status
フォーカル・ポイントの状況。 これは、以下のいずれかです。

アクティブ （ アクティブ ）
フォーカル・ポイントは現在活動状態です。

追加がアクティブではない
フォーカル・ポイントは現在アクティブではありません。

追加保留中
フォーカル・ポイントは活動保留中です。 これは、暗黙要求がフォーカル・ポイントに送信され、応答が受信される前に発生します。

非 NEVER_ACTIVE
指定されたカテゴリーのアプリケーション登録が受け入れられたが、指定されたカテゴリーについてのフォーカル・ポイント情報はありません。

fp_data.fp_ルーティング
アプリケーションがフォーカル・ポイントにトラフィックを経路指定するためにデフォルトまたは直接ルーティングを使用するかどうかを指定 これは、以下のいずれかです。

デフォルト （ デフォルト ）
MDS_MU は、デフォルトのルーティングを使用してフォーカル・ポイントに送達される必要があります。

間接
MDS_MU は、セッションで直接フォーカル・ポイントに経路指定される必要があります。

アプリケーションの *fp_data.number_of_appls*
このフォーカル・ポイント・カテゴリーに登録されているアプリケーションの数。

アプリケーション名
フォーカル・ポイント・カテゴリーに登録されたアプリケーションの名前。 これは、 システム・ネットワーク体系: 管理サービス参照 に指定されている MS 規律特定アプリケーション・プログラム (Bibliography を参照) 、またはタイプ 1134 文字を使用する EBCDIC ストリングのいずれかです。名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_MS_CATEGORY
The *list_options* parameter was set to AP_LIST_INCLUSIVE, to list all entries starting from the supplied name, but the *ms_category* parameter was not valid.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: 関数はサポートされません

ローカル・ノード構成がそれをサポートしていないために verb が正常に実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc

付加機能がサポートされていません
ローカル・ノードは MS ネットワーク管理機能をサポートしていません。これは、 DEFINE_NODE verb の *mds_supported* パラメーターによって定義されます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

クエリー・グローバル・ログ・タイプ

この verb を使用すると、 NOF アプリケーションは、 CS Linux レコードがログ・ファイルに記録する情報のタイプを判別できます。 すべてのサーバーで使用されるデフォルト値 (SET_LOG_TYPE によって特定のサーバー上でオーバーライドされる場合を除く) を指定します。 QUERY_LOG_TYPE は、特定のサーバーで使用されている値を判別するために使用できます。

CS Linux は、以下のタイプのイベントに関するメッセージをログに記録

問題
ユーザー (セッションの異常終了など) に認識できないようにシステムを機能低下させる異常イベント。

例外
システムを低下させる可能性があるが、即時にユーザーに認識できない異常イベント (リモート・システムからではないメッセージの 受信など) 。

監査
通常のイベント (セッションの開始など) 。

問題メッセージおよび例外メッセージは、エラー・ログ・ファイルに記録されます。監査メッセージは監査ログ・ファイルに記録されます。 問題メッセージは常にログに記録され、使用不可にすることはできませんが、他の 2 つのメッセージ・タイプをそれぞれログに記録するかどうか 2 つのファイル (監査およびエラー) のそれぞれについて、簡潔なロギングを使用するか (メッセージのテキストとメッセージ・ソースの要約のみを含む) 、またはフル・ロギング (メッセージ・ソースの完全な詳細、原因、および必要なアクションを含む) を指定することができます。

この verb は、現在中央ロガーとして動作しているノードに対して発行する必要があります。詳細については、53 ページの『接続ノード』を参照してください。

VCB structure

```
typedef struct query_global_log_type
{
   AP_UINT16       opcode;            /* verb operation code                */
   unsigned char   reserv2;           /* reserved                           */
   unsigned char   format;            /* reserved                           */
   AP_UINT16       primary_rc;        /* primary return code                */
   AP_UINT32       secondary_rc;      /* secondary return code              */
   unsigned char   audit;             /* audit logging on or off            */
   unsigned char   exception;         /* exception logging on or off        */
   unsigned char   succinct_audits;   /* use succinct logging in audit file? */
   unsigned char   succinct_errors;   /* use succinct logging in error file? */
   unsigned char   reserv3[4];        /* reserved                           */
} QUERY_GLOBAL_LOG_TYPE;
```

Supplied parameters

The application supplies the following parameter:

opcode
AP_QUERY_GLOBAL_LOG_TYPE

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

secondary_rc
Not used.

audit
This parameter indicates whether audit messages are recorded. Possible values are:

AP_YES
Audit messages are recorded.

AP_NO
Audit messages are not recorded.

exception
This parameter indicates whether exception messages are recorded. Possible values are:

AP_YES
Exception messages are recorded.

AP_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:

AP_YES
Succinct logging: 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 of the cause of the log and any action required, you can use the `snahelp` utility.

AP_NO
Full logging: each message in the log file includes a full listing of the message header information, the message text string and parameters, and additional information about the cause of the log and any action required.

If you are using central logging, the choice of succinct or full logging for messages from all computers is determined by the setting of this parameter on the server acting as the central logger; this setting may either be from the SET_GLOBAL_LOG_TYPE verb, or from a SET_LOG_TYPE verb issued to that server to override the default.

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. The possible values and their meanings are the same as for the *succinct_audits* parameter.

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc

AP_NOT_CENTRAL_LOGGER
The verb was issued to a node that is not the central logger.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

QUERY_ISR_SESSION

QUERY_ISR_SESSION returns list information about the sessions for which a network node is providing intermediate session routing.

This verb can be used to obtain information about a specific session, or about a number of sessions, depending on the options used. It can be used only if the CS Linux node is a network node, and is not valid if it is an end node or LEN node.

This list is ordered by *fqpcid.pcid* first and then by EBCDIC lexicographical ordering on *fqpcid.fqcp_name*. The format of the `fqpcid` structure is an 8-byte PCID (Procedure Correlator Identifier) and the network qualified CP name of the session originator.

This verb must be issued to a running node.

VCB structure

```
typedef struct query_isr_session
{
   AP_UINT16       opcode;              /* verb operation code              */
   unsigned char   reserv2;             /* reserved                         */
   unsigned char   format;              /* reserved                         */
   AP_UINT16       primary_rc;          /* primary return code              */
   AP_UINT32       secondary_rc;        /* secondary return code            */
   unsigned char   *buf_ptr;            /* pointer to buffer                */
   AP_UINT32       buf_size;            /* buffer size                      */
   AP_UINT32       total_buf_size;      /* total buffer size required       */
   AP_UINT16       num_entries;         /* number of entries                */
   AP_UINT16       total_num_entries;   /* total number of entries          */
   unsigned char   list_options;        /* listing options                  */
   unsigned char   session_type;        /* is this query for DLUR or regular*/
                                        /*   ISR sessions?                  */
```

```
   FQPCID          fqpcid;              /* fully qualified procedure        */
                                        /* correlator ID                    */
} QUERY_ISR_SESSION;
```

```
typedef struct isr_session_summary
{
   AP_UINT16       overlay_size;        /* size of returned entry           */
   FQPCID          fqpcid;              /* fully qualified procedure        */
                                        /* correlator ID                    */
} ISR_SESSION_SUMMARY;
```

```
typedef struct isr_session_detail
{
   AP_UINT16       overlay_size;        /* size of returned entry            */
   AP_UINT16       sub_overlay_size;    /* offset to appended RSCV           */
   FQPCID          fqpcid;              /* fully qualified procedure         */
                                        /*  correlator ID                    */
   unsigned char   trans_pri;           /* Transmission priority:            */
   unsigned char   cos_name[8];         /* Class of Service name             */
   unsigned char   ltd_res;             /* Session spans a limited resource  */
   unsigned char   reserv1[2];          /* reserved                          */
   EXTENDED_SESSION_STATS  pri_ext_sess_stats; /* primary hop session stats   */
   EXTENDED_SESSION_STATS  sec_ext_sess_stats; /* secondary hop session stats */
   unsigned char   sess_lu_type;        /* session LU type                   */
   unsigned char   sess_lu_level;       /* session LU level                  */
   unsigned char   pri_tg_number;       /* Primary session TG number         */
   unsigned char   sec_tg_number;       /* Secondary session TG number       */
   AP_UINT32       rtp_tcid;            /* RTP TC identifier                 */
   AP_UINT32       time_active;         /* time elapsed since activation     */
   unsigned char   isr_state;           /* current state of ISR session      */
   unsigned char   reserv2[11];         /* reserved                          */
   unsigned char   mode_name[8];        /* mode name                         */
   unsigned char   pri_lu_name[17];     /* primary LU name                   */
   unsigned char   sec_lu_name[17];     /* secondary LU name                 */
   unsigned char   pri_adj_cp_name[17]; /* primary stage adjacent CP name    */
   unsigned char   sec_adj_cp_name[17]; /* secondary stage adjacent CP name  */
   unsigned char   reserv3[3];          /* reserved                          */
   unsigned char   rscv_len;            /* length of following RSCV          */
} ISR_SESSION_DETAIL;
```

The ISR session detail structure may be followed by a 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 RSCVs should be stored for ISR sessions.

```
typedef struct fqpcid
{
   unsigned char   pcid[8];             /* procedure correlator identifier  */
   unsigned char   fqcp_name[17];       /* originator's network qualified   */
                                        /*  CP name                         */
   unsigned char   reserve3[3];         /* reserved                         */
} FQPCID;
```

```
typedef struct extended_session_stats
{
   AP_UINT16       rcv_ru_size;          /* session receive RU size          */
   AP_UINT16       send_ru_size;         /* session send RU size             */
   AP_UINT16       max_send_btu_size;    /* maximum send BTU size            */
   AP_UINT16       max_rcv_btu_size;     /* maximum rcv BTU size             */
   AP_UINT16       max_send_pac_win;     /* maximum send pacing window size  */
   AP_UINT16       cur_send_pac_win;     /* current send pacing window size  */
   AP_UINT16       send_rpc;             /* send residual pacing count       */
   AP_UINT16       max_rcv_pac_win;      /* maximum rcv pacing window size   */
   AP_UINT16       cur_rcv_pac_win;      /* current rcv pacing window size   */
   AP_UINT16       rcv_rpc;              /* receive residual pacing count    */
   AP_UINT32       send_data_frames;     /* number of data frames sent       */
   AP_UINT32       send_fmd_data_frames; /* num fmd data frames sent         */
   AP_UINT32       send_data_bytes;      /* number of data bytes sent        */
   AP_UINT32       send_fmd_data_bytes;  /* number of fmd data bytes sent    */
   AP_UINT32       rcv_data_frames;      /* number of data frames received   */
   AP_UINT32       rcv_fmd_data_frames;  /* num fmd data frames received     */
   AP_UINT32       rcv_data_bytes;       /* number of data bytes received    */
   AP_UINT32       rcv_fmd_data_bytes;   /* number of fmd data bytes received */
   unsigned char   sidh;                 /* session ID high byte (from LFSID) */
   unsigned char   sidl;                 /* session ID low byte (from LFSID)  */
   unsigned char   odai;                 /* ODAI bit set                      */
```

```
    unsigned char   ls_name[8];          /* link station name                */
    unsigned char   pacing_type;         /* type of pacing in use            */
    unsigned char   reserv1[100];        /* reserved                         */
} EXTENDED_SESSION_STATS;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_ISR_SESSION

buf_ptr
A pointer to a data buffer that CS Linux will use to return the requested information.

buf_size
Size of the supplied data buffer.

num_entries
Maximum number of sessions for which data should be returned. To request data for a specific session rather than a range, specify the value 1. To return as many entries as possible, specify zero; in this case, CS Linux will return the maximum number of entries that can be accommodated in the supplied data buffer.

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

AP_SUMMARY
Summary information only.

AP_DETAIL
Detailed information.

Combine this value using a logical OR operation with one of the following values:

AP_FIRST_IN_LIST
Start at the first entry in the list.

AP_LIST_INCLUSIVE
Start at the entry specified by the *pcid* and *fqcp_name* parameters.

AP_LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *pcid* and *fqcp_name* parameters.

The list is ordered by *pcid* (numerically), and then by *fqcp_name*. For more information about how the application can obtain specific entries from the list, see “List options for QUERY_* Verbs” on page 34.

session_type
Specifies whether DLUR-maintained sessions or regular ISR sessions are being queried. Possible values are:

AP_DLUR_SESSIONS
DLUR-maintained sessions are being queried.

AP_ISR_SESSIONS
Regular ISR sessions are being queried.

fqpcid.pcid
Procedure Correlator ID. This is an 8-byte hexadecimal string. This value is ignored if *list_options* is set to AP_FIRST_IN_LIST.

fqpcid.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 AP_FIRST_IN_LIST.

The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

isr_session_summary.overlay_size
The size of the returned `isr_session_summary` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `isr_session_summary` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

isr_session_summary.fqpcid.pcid
Procedure Correlator ID.

isr_session_summary.fqpcid.fqcp_name
Fully qualified CP name. The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

isr_session_detail.overlay_size
The size of the returned `isr_session_detail` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `isr_session_detail` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

isr_session_detail.fqpcid.pcid
Procedure Correlator ID.

isr_session_detail.fqpcid.fqcp_name
Fully qualified CP name. The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

isr_session_detail.trans_pri
Transmission priority. This parameter has one of the following values:

- AP_LOW AP_MEDIUM
- AP_HIGH AP_NETWORK

isr_session_detail.cos_name
Class of service name. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces.

isr_session_detail.ltd_res
Specifies whether the session uses a limited resource link. Possible values are:

AP_YES
Session uses a limited resource link.

AP_NO
Session does not use a limited resource link.

For each of the two sessions (primary and secondary), the extended_session_stats structure contains the following fields, each preceded by *isr_session_detail.pri_ext_sess_stats.*_** for the primary session and *isr_session_detail.sec_ext_sess_stats.*_** for the secondary session:

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.

pacing_type
Receive pacing type in use on the session. Possible values are:

- AP_NONE
- AP_PACING_FIXED
- AP_PACING_ADAPTIVE

The following parameters are also returned (these parameters are not part of the session_stats structure):

isr_session.detail.sess_lu_type
The LU type of the session specified on the BIND. Possible values are (LU type 5 is intentionally omitted):

- AP_LU_TYPE_0
- AP_LU_TYPE_1
- AP_LU_TYPE_2
- AP_LU_TYPE_3
- AP_LU_TYPE_4
- AP_LU_TYPE_6
- AP_LU_TYPE_7
- AP_LU_TYPE_UNKNOWN

isr_session.detail.sess_lu_level
The LU level of the session. Possible values are:

- AP_LU_LEVEL_0
- AP_LU_LEVEL_1
- AP_LU_LEVEL_2
- AP_LU_LEVEL_UNKNOWN

For LU types other than 6, this parameter is set to AP_LU_LEVEL_0. The value AP_LU_LEVEL_UNKNOWN is always returned unless collection of names has been enabled using DEFINE_ISR_STATS.

isr_session.detail.pri_tg_number
The TG number associated with the link traversed by the primary session hop. If the primary session stage traverses an RTP connection, zero is returned. The value zero is always returned unless collection of names has been enabled using DEFINE_ISR_STATS.

isr_session.detail.sec_tg_number
The TG number associated with the link traversed by the secondary session hop. If the secondary session stage traverses an RTP connection, zero is returned. The value zero is always returned unless collection of names has been enabled using DEFINE_ISR_STATS.

isr_session.detail.rtp_tcid
Total TC ID for the RTP connection. This is returned in cases where this ISR session forms part of an ANR/ISR boundary. In other cases, this parameter is set to zero. The value zero is always returned unless collection of names has been enabled using DEFINE_ISR_STATS.

isr_session.detail.time_active
The elapsed time since the activation of the session, in hundredths of a second. The value zero is always returned unless collection of names has been enabled using DEFINE_ISR_STATS.

isr_session.detail.isr_state
The current state of the session. Possible values are:

- `AP_ISR_INACTIVE`
- `AP_ISR_PENDING_ACTIVE`
- `AP_ISR_ACTIVE`
- `AP_ISR_PENDING_INACTIVE`

isr_session.detail.mode_name
The mode name for the session. This is an 8-byte alphanumeric type-A EBCDIC string starting with a letter, padded on the right with EBCDIC spaces. All binary zeros are returned unless collection of names has been enabled using DEFINE_ISR_STATS.

isr_session.detail.pri_lu_name
The primary LU name of the session. This name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. The name consists of a network ID of 1-8 A-string characters, an EBCDIC dot (period) character, and an LU name of 1-8 A-string characters. If this name is not available, all binary zeros are returned in this field. All binary zeros are always returned unless collection of names has been enabled using DEFINE_ISR_STATS.

isr_session.detail.sec_lu_name
The secondary LU name of the session. This name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. The name consists of a network ID of 1-8 A-string characters, an EBCDIC dot (period) character, and an LU name of 1-8 A-string characters. If this name is not available, all binary zeros are returned in this field. All binary zeros are always returned unless collection of names has been enabled using DEFINE_ISR_STATS.

isr_session.detail.pri_adj_cp_name
The primary stage adjacent CP name of this session. If the primary session traverses an RTP connection, the CP name of the remote RTP endpoint is returned. This name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. The name consists of a network ID of 1-8 A-string characters, an EBCDIC dot (period) character, and a CP name of 1-8 A-string characters. If this name is not available, all binary zeros are returned in this field. All binary zeros are always returned unless collection of names has been enabled using DEFINE_ISR_STATS.

isr_session.detail.sec_adj_cp_name
The secondary stage adjacent CP name of this session. If the secondary session traverses an RTP connection, the CP name of the remote RTP endpoint is returned. This name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. The name consists of a network ID of 1-8 A-string characters, an EBCDIC dot (period) character, and a CP name of 1-8 A-string characters. If this name is not available, all binary zeros are returned in this field. All binary zeros are always returned unless collection of names has been enabled using DEFINE_ISR_STATS.

isr_session_detail.rscv_len
Length of the RSCV which is appended to the `session_detail` structure. (If none is appended, then the length is zero.)

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_FQPCID
The *list_options* parameter was set to AP_LIST_INCLUSIVE to list all entries starting from the supplied name, but the *pcid* parameter was not valid.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: 関数はサポートされません

ローカル・ノードがネットワーク・ノードではないために verb が正常に実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc

エイプ・ インバリデータ
ローカル・ノードがネットワーク・ノードではありません。 この verb は、ネットワーク・ノードでのみ使用できます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

QUERY_KERNEL_MEMORY_LIMIT

This verb returns information about the amount of kernel memory that CS Linux is currently using, the maximum amount it has used, and the configured limit. This allows you to check memory usage and set the limit appropriately, to ensure that sufficient memory is available for CS Linux components and for other programs on the Linux computer.

You can specify the kernel memory limit when starting the CS Linux software (for more information, see the *IBM Communications Server for Data Center Deployment on Linux Administration Guide*), or modify it later when the node is running (using the SET_KERNEL_MEMORY_LIMIT verb).

VCB 構造体

```
typedef struct query_kernel_memory_limit
{
   AP_UINT16        opcode;            /* verb operation code                */
   unsigned char    reserv2;           /* reserved                           */
   unsigned char    format;            /* reserved                           */
   AP_UINT16        primary_rc;        /* primary return code                */
   AP_UINT32        secondary_rc;      /* secondary return code              */
   AP_UINT32        limit;             /* kernel memory limit, 0 => no limit */
   AP_UINT32        actual;            /* current amount of memory allocated */
   AP_UINT32        max_used;          /* maximum amount of memory allocated */
   unsigned char    reset_max_used;    /* set max_used = actual              */
   unsigned char    reserv3[8];        /* Reserved                           */
} QUERY_KERNEL_MEMORY_LIMIT;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_KERNEL_MEMORY_LIMIT

reset_max_used
Specify whether CS Linux should reset the *max_used* value (after returning it on this verb) to match the amount of memory currently allocated. This ensures that a subsequent QUERY_KERNEL_MEMORY_LIMIT verb will return the maximum amount used since this verb, rather than the maximum amount used since the system was started (or since the *max_used* value was last reset). Possible values are:

AP_YES
Reset the *max_used* value to match the current memory allocation.

AP_NO
Do not reset the *max_used* value.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

secondary_rc
Not used.

limit
The maximum amount of kernel memory, in bytes, that CS Linux is permitted to use at any time. If a CS Linux component attempts to allocate kernel memory that would take the total amount of memory currently allocated to CS Linux components above this limit, the allocation attempt will fail. A value of zero indicates no limit.

actual
The amount of kernel memory, in bytes, currently allocated to CS Linux components.

max_used
The maximum amount of kernel memory, in bytes, that has been allocated to CS Linux components at any time since the *max_used* parameter was last reset (as described for *reset_max_used* above), or since the CS Linux software was started.

reset_max_used
Specifies whether CS 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 verb 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:

AP_YES
CS Linux resets the *max_used* value to match the current memory allocation.

AP_NO
CS Linux does not reset the *max_used* value.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

QUERY_LOCAL_LU

QUERY_LOCAL_LU returns information about local LUs.

This verb can be used to obtain either 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).

VCB 構造体

```
typedef struct query_local_lu
{
   AP_UINT16          opcode;               /* verb operation code          */
   unsigned char      reserv2;              /* reserved                     */
   unsigned char      format;               /* reserved                     */
   AP_UINT16          primary_rc;           /* primary return code          */
   AP_UINT32          secondary_rc;         /* secondary return code        */
   unsigned char      *buf_ptr;             /* pointer to buffer            */
   AP_UINT32          buf_size;             /* buffer size                  */
   AP_UINT32          total_buf_size;       /* total buffer size required   */
   AP_UINT16          num_entries;          /* number of entries            */
   AP_UINT16          total_num_entries;    /* total number of entries      */
   unsigned char      list_options;         /* listing options              */
   unsigned char      reserv3;              /* reserved                     */
   unsigned char      lu_name[8];           /* LU name                      */
   unsigned char      lu_alias[8];          /* LU alias                     */
   unsigned char      pu_name[8];           /* PU name filter               */
} QUERY_LOCAL_LU;
```

```
typedef struct local_lu_summary
{
   AP_UINT16          overlay_size;         /* size of returned entry       */
   unsigned char      lu_name[8];           /* LU name                      */
   unsigned char      lu_alias[8];          /* LU alias                     */
   unsigned char      description[32];      /* resource description         */
   unsigned char      reserv1[16];          /* reserved                     */
} LOCAL_LU_SUMMARY;
```

```
typedef struct local_lu_detail
{
   AP_UINT16          overlay_size;         /* size of returned entry       */
   unsigned char      lu_name[8];           /* LU name                      */
   LOCAL_LU_DEF_DATA  def_data;             /* defined data                 */
   LOCAL_LU_DET_DATA  det_data;             /* determined data              */
} LOCAL_LU_DETAIL;
```

```
typedef struct local_lu_def_data
{
   unsigned char    description[32];       /* resource description         */
   unsigned char    reserv1;               /* reserved                     */
   unsigned char    security_list_name[14];   /* security access list name */
   unsigned char    reserv3;               /* reserved                     */
   unsigned char    lu_alias[8];           /* local LU alias               */
   unsigned char    nau_address;           /* NAU address                  */
   unsigned char    syncpt_support;        /* is Syncpoint supported?      */
   AP_UINT16        lu_session_limit;      /* LU session limit             */
   unsigned char    default_pool;          /* is LU in the pool of default */
                                           /*  LUs?                        */
   unsigned char    reserv2;               /* reserved                     */
   unsigned char    pu_name[8];            /* PU name                      */
   unsigned char    lu_attributes;         /* LU attributes                */
   unsigned char    sscp_id[6]             /* SSCP ID                      */
   unsigned char    disable;               /* disable or enable local LU   */
   ROUTING_DATA     attach_routing_data;   /* routing data for incoming    */
                                           /* attaches                     */
   unsigned char    reserv6;               /* reserved                     */
   unsigned char    reserv4[7];            /* reserved                     */
   unsigned char    reserv5[16];           /* reserved                     */
} LOCAL_LU_DEF_DATA;
```

```
typedef struct local_lu_det_data
{
   unsigned char      lu_sscp_sess_active;   /* Is LU-SSCP session active  */
   unsigned char      appl_conn_active;      /* reserved                   */
   unsigned char      reserv1[2];            /* reserved                   */
   SESSION_STATS      lu_sscp_stats;         /* LU-SSCP session statistics */
   unsigned char      sscp_id[6];            /* SSCP ID                    */
} LOCAL_LU_DET_DATA;
```

```
typedef struct session_stats
{
   AP_UINT16          rcv_ru_size;           /* session receive RU size      */
   AP_UINT16          send_ru_size;          /* session send Ru size         */
   AP_UINT16          max_send_btu_size;     /* max send BTU size            */
   AP_UINT16          max_rcv_btu_size;      /* max rcv BTU size             */
   AP_UINT16          max_send_pac_win;      /* max send pacing window size  */
   AP_UINT16          cur_send_pac_win;      /* current send pacing win size */
   AP_UINT16          max_rcv_pac_win;       /* max receive pacing win size  */
   AP_UINT16          cur_rcv_pac_win;       /* current receive pacing       */
                                             /*  window size                 */
   AP_UINT32          send_data_frames;      /* number of data frames sent   */
   AP_UINT32          send_fmd_data_frames;  /* num of fmd data frames sent  */
   AP_UINT32          send_data_bytes;       /* number of data bytes sent    */
   AP_UINT32          rcv_data_frames;       /* num data frames received     */
   AP_UINT32          rcv_fmd_data_frames;   /* num of fmd data frames recvd */
   AP_UINT32          rcv_data_bytes;        /* number of data bytes received*/
   unsigned char      sidh;                  /* session ID high byte         */
   unsigned char      sidl;                  /* session ID low byte          */
   unsigned char      odai;                  /* ODAI bit set                 */
   unsigned char      ls_name;               /* link station name            */
   unsigned char      pacing_type;           /* type of pacing in use        */
} SESSION_STATS;
```

```
typedef struct routing_data
{
   unsigned char      sys_name[128];         /* Name of target system for TP */
   AP_INT32           timeout;               /* timeout value in seconds     */
   unsigned char      back_level;            /* reserved                     */
   unsigned char      reserved[59];          /* reserved                     */
} ROUTING_DATA;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_LOCAL_LU

buf_ptr
A pointer to a data buffer that CS Linux will use to return the requested information.

buf_size
Size of the supplied data buffer.

num_entries
Maximum number of LUs for which data should be returned. To request data for a specific LU rather than a range, specify the value 1. To return as many entries as possible, specify zero; in this case, CS Linux will return the maximum number of entries that can be accommodated in the supplied data buffer.

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

AP_SUMMARY
Summary information only.

AP_DETAIL
Detailed information.

Combine this value using a logical OR operation with one of the following values:

AP_FIRST_IN_LIST
Start at the first entry in the list.

AP_LIST_INCLUSIVE
Start at the entry specified by the *lu_name* or *lu_alias* parameter.

AP_LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *lu_name* or *lu_alias* parameter.

If AP_FIRST_IN_LIST is specified, you can also include the following option, using a logical OR operation:

AP_LIST_BY_ALIAS
The list is returned in order of LU alias rather than LU name. This option is only valid if AP_FIRST_IN_LIST is also specified. (For AP_LIST_FROM_NEXT or AP_LIST_INCLUSIVE, the list is in order of LU alias or LU name, depending on which was specified as the index into the list.)

For more information about how the application can obtain specific entries from the list, see "List options for QUERY_* Verbs" on page 34. The list is in EBCDIC lexicographical order (irrespective of the length of each name).

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 AP_FIRST_IN_LIST. To identify the LU by its alias instead of its name, set this parameter to 8 binary zeros, and specify the alias in the *lu_alias* parameter; to identify the default LU, set both *lu_name* and *lu_alias* to 8 binary zeros.

The name is an 8-byte EBCDIC string, padded on the right with EBCDIC spaces if the name is shorter than 8 characters.

lu_alias
LU 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 AP_FIRST_IN_LIST.

This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 characters. To identify the LU by its LU name instead of its alias, set this parameter to 8 binary zeros, and specify the name in the *lu_name* parameter; to identify the default LU, set both *lu_name* and *lu_alias* to 8 binary zeros.

pu_name
PU name filter. To return information only on LUs associated with a specific PU, specify the PU name; to return information without filtering on PU name, set this parameter to 8 binary zeros.

The name is an 8-byte type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

local_lu_summary.overlay_size
The size of the returned local_lu_summary structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `local_lu_summary` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

local_lu_summary.lu_name
LU name. This name is an 8-byte type-A EBCDIC character string.

local_lu_summary.lu_alias
LU alias. This is an 8-byte ASCII character string.

local_lu_summary.description
A null-terminated text string describing the local LU, as specified in the definition of the LU.

local_lu_detail.overlay_size
The size of the returned `local_lu_detail` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `local_lu_detail` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

local_lu_detail.lu_name
LU name. This name is an 8-byte type-A EBCDIC character string.

local_lu_detail.def_data.description
A null-terminated text string describing the local LU, as specified in the definition of the LU.

local_lu_detail.def_data.security_list_name
Name of the security access list used by this local LU (defined using the DEFINE_SECURITY_ACCESS_LIST verb). If this parameter is set to 14 binary zeros, the LU is available for use by any user.

local_lu_detail.def_data.lu_alias
LU alias. This is an 8-byte ASCII character string.

local_lu_detail.def_data.nau_address
Network accessible unit address of the LU. This is in the range 1-255 if the LU is a dependent LU, or zero if the LU is an independent LU.

local_lu_detail.def_data.syncpt_support
Specifies whether the LU supports Syncpoint functions. Possible values are:

AP_YES
Syncpoint is supported.

AP_NO
Syncpoint is not supported.

local_lu_detail.def_data.lu_session_limit
Maximum total number of sessions (across all modes) for the local LU. A value of zero indicates that there is no limit.

local_lu_detail.def_data.default_pool
Specifies whether the LU is in the pool of default dependent LUs. When an application attempts to start a conversation without specifying a local LU name, CS Linux will select an unused LU from this pool. Possible values are:

AP_YES
The LU is in the pool of default LUs, and can be used by applications that do not specify an LU name.

AP_NO
The LU is not in the pool.

If the LU is an independent LU, this parameter is reserved.

local_lu_detail.def_data.pu_name
For dependent LUs, this parameter identifies the PU that this LU will use. This is an 8-byte type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces if necessary. For independent LUs, this field is not used; it is set to 8 binary zeros.

local_lu_detail.def_data.lu_attributes
Configured LU attributes. Possible values are:

AP_NONE
No additional information identified.

AP_DISABLE_PWSUB
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. CS 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). The option should not be set unless there are problems with the remote system's implementation of password substitution.

local_lu_detail.def_data.sscp_id
Specifies the ID of the SSCP permitted to activate this LU. It is a 6-byte binary field. This parameter is used only by dependent LUs, and is set to all binary zeros for independent LUs or if the LU can be activated by any SSCP.

local_lu_detail.def_data.attach_routing_data.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. This identifies the computer where the target TP runs.

If this parameter is set to binary zeros, CS Linux routes the incoming Allocate request dynamically to a running copy of the TP, if available, or attempts to start the TP on the same computer as the local LU.

local_lu_detail.def_data.attach_routing_data.timeout
The timeout value (in seconds) for dynamic load requests. A request will time out if the invoked TP has not issued a Receive_Allocate verb (APPC), or Accept_Conversation or Accept_Incoming (CPI-C), within this time. A value of -1 indicates no timeout (dynamic load requests will wait indefinitely).

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 verb for the appropriate session between this LU and the partner LU.

local_lu_detail.det_data.lu_sscp_session_active
Specifies whether the LU-SSCP session is active. Possible values are:

AP_YES
The LU-SSCP session is active.

AP_NO
The LU-SSCP session is not active.

local_lu_detail.det_data.lu_sscp_stats
Statistics for the LU-SSCP session.

local_lu_detail.det_data.lu_sscp_stats.rcv_ru_size
This parameter is always reserved.

local_lu_detail.det_data.lu_sscp_stats.send_ru_size
This parameter is always reserved.

local_lu_detail.det_data.lu_sscp_stats.max_send_btu_size
Maximum basic transmission unit (BTU) size that can be sent.

local_lu_detail.det_data.lu_sscp_stats.max_rcv_btu_size
Maximum BTU size that can be received.

local_lu_detail.det_data.lu_sscp_stats.max_send_pac_win
This parameter is always set to zero.

local_lu_detail.det_data.lu_sscp_stats.cur_send_pac_win
This parameter is always set to zero.

local_lu_detail.det_data.lu_sscp_stats.max_rcv_pac_win
This parameter is always set to zero.

local_lu_detail.det_data.lu_sscp_stats.cur_rcv_pac_win
This parameter is always set to zero.

local_lu_detail.det_data.lu_sscp_stats.send_data_frames
Number of normal flow data frames sent

local_lu_detail.det_data.lu_sscp_stats.send_fmd_data_frames
Number of normal flow function management data (FMD) frames sent.

local_lu_detail.det_data.lu_sscp_stats.send_data_bytes
Number of normal flow data bytes sent.

local_lu_detail.det_data.lu_sscp_stats.rcv_data_frames
Number of normal flow data frames received.

local_lu_detail.det_data.lu_sscp_stats.rcv_fmd_data_frames
Number of normal flow FMD data frames received.

local_lu_detail.det_data.lu_sscp_stats.rcv_data_bytes
Number of normal flow data bytes received.

local_lu_detail.det_data.lu_sscp_stats.sidh
Session ID high byte.

local_lu_detail.det_data.lu_sscp_stats.sidl
Session ID low byte.

local_lu_detail.det_data.lu_sscp_stats.odai
Origin Destination Assignor Indicator. When bringing up a session, the sender of the ACTLU sets this parameter to zero if the local node contains the primary link station, and sets it to one if the ACTLU sender is the node containing the secondary link station.

local_lu_detail.det_data.lu_sscp_stats.ls_name
Link station name associated with the statistics This is an 8-byte string in a locally displayable character set. All eight bytes are significant. This parameter can be used to correlate this session with the link over which the session flows.

local_lu_detail.det_data.lu_sscp_stats.pacing_type
Receive pacing type in use on the LU-SSCP session. This parameter is set to AP_NONE.

local_lu_detail.det_data.sscp_id
This parameter is a 6-byte field containing the SSCP ID received in the ACTPU for the PU used by this LU.

This parameter is reserved if *lu_sscp_sess_active* is not set to AP_YES.

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

エイプ無効ルリエイリアス
list_options パラメーターが **リストを含む** （ **包括的** ）に設定され、指定された名前から始まるすべての項目がリストされましたが、 *lu_alias* パラメーターが無効でした。

ファイル名の変更
list_options パラメーターが **リストを含む** （ **包括的** ）に設定され、指定された名前から始まるすべての項目がリストされましたが、 *lu_name* パラメーターが無効でした。

ファイルの追加リスト・オプション
list_options パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

クエリー・ローカル・トポロジー

すべての APPN ノードは、すべての隣接ノードに対する TG に関する情報を保持するローカル・トポロジー・データベースを保守します。 QUERY_LOCAL_TOPOLOGY により、これらの TG についての情報を戻すことができます。

この verb は、使用されるオプションに応じて、特定の TG または複数の TG に関する要約情報または詳細情報のいずれかを取得するために使用できます。

この verb は実行中のノードに対して発行する必要があります

VCB 構造体

```
typedef struct query_local_topology
{
   AP_UINT16          opcode;                /* verb operation code         */
   unsigned char      reserv2;               /* reserved                    */
   unsigned char      format;                /* reserved                    */
   AP_UINT16          primary_rc;            /* primary return code         */
   AP_UINT32          secondary_rc;          /* secondary return code       */
   unsigned char      *buf_ptr;              /* pointer to buffer           */
   AP_UINT32          buf_size;              /* buffer size                 */
   AP_UINT32          total_buf_size;        /* total buffer size required  */
   AP_UINT16          num_entries;           /* number of entries           */
   AP_UINT16          total_num_entries;     /* total number of entries     */
   unsigned char      list_options;          /* listing options             */
   unsigned char      reserv3;               /* reserved                    */
   unsigned char      dest[17];              /* TG destination node         */
   unsigned char      dest_type;             /* TG destination node type    */
   unsigned char      tg_num;                /* TG number                   */
} QUERY_LOCAL_TOPOLOGY;
```

```
typedef struct local_topology_summary
{
   AP_UINT16          overlay_size;          /* size of returned entry      */
   unsigned char      dest[17];              /* TG destination node         */
   unsigned char      dest_type;             /* TG destination node type    */
   unsigned char      tg_num;                /* TG number                   */
} LOCAL_TOPOLOGY_SUMMARY;
```

```
typedef struct local_topology_detail
{
   AP_UINT16          overlay_size;          /* size of returned entry      */
   unsigned char      dest[17];              /* TG destination node         */
   unsigned char      dest_type;             /* TG destination node type    */
   unsigned char      tg_num;                /* TG number                   */
   unsigned char      reserv1;               /* reserved                    */
   LINK_ADDRESS       dlc_data;              /* DLC signalling data         */
   AP_UINT32          rsn;                   /* resource sequence number    */
   unsigned char      status;                /* tg status                   */
   TG_DEFINED_CHARS   tg_chars;              /* TG characteristics          */
   unsigned char      cp_cp_session_active;  /* CP-CP sessions active?      */
   unsigned char      branch_link_type;      /* Up or down link?            */
   unsigned char      branch_tg;             /* Branch TG?                  */
   unsigned char      appended_data_format;  /* Format of appended data     */
```

```
   unsigned char      appended_data_len;      /* Length of appended data     */
   unsigned char      reserva[11];            /* reserved                    */
} LOCAL_TOPOLOGY_DETAIL;
```

```
typedef struct link_address
{
  unsigned char   format;                /* type of link address          */
  unsigned char   reserve1;              /* reserved                      */
  AP_UINT16       length;                /* length                        */
  unsigned char   address[32];           /* address                       */
} LINK_ADDRESS;
```

TG_DEFINED_CHARS ストラクチャーの詳細については、104 ページの『DEFINE_LS』を参照してください。

list_options パラメーターに詳細情報が指定されている場合は、戻された情報に TG 記述子 CV が付加される場合があります。 詳しくは、パラメーター ローカル・トポロジー詳細 *. appended_data_format* および ローカル・トポロジー詳細 *. appended_data_len* の説明を参照してください。

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_LOCAL_TOPOLOGY

buf_ptr
A pointer to a data buffer that CS Linux will use to return the requested information.

buf_size
Size of the supplied data buffer.

num_entries
Maximum number of entries for which data should be returned. To request a specific entry rather than a range, specify the value 1. To return as many entries as possible, specify zero; in this case, CS Linux will return the maximum number of entries that can be accommodated in the supplied data buffer.

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

AP_SUMMARY
Summary information only.

AP_DETAIL
Detailed information.

Combine this value using a logical OR operation with one of the following values:

AP_FIRST_IN_LIST
Start at the first entry in the list.

AP_LIST_INCLUSIVE
Start at the entry specified by the combination of the *dest*, *dest_type*, and *tg_num* parameters.

AP_LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the combination of the *dest*, *dest_type*, and *tg_num* parameters.

The list is ordered by *dest*, then by *dest_type* (in the order AP_NETWORK_NODE, AP_END_NODE, AP_VRN), and lastly in numerical order of *tg_num*. For more information about how the list is ordered and how the application can obtain specific entries from it, see “List options for QUERY_* Verbs” on page 34.

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 AP_FIRST_IN_LIST.

The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

dest_type
Node type of the destination node for this TG. This value is ignored if *list_options* is set to AP_FIRST_IN_LIST. Possible values are:

AP_NETWORK_NODE
Network node.

AP_VRN
Virtual routing node.

AP_END_NODE
End node or LEN node.

AP_LEARN_NODE
Unknown node type.

tg_num
Number associated with the TG. This value is ignored if *list_options* is set to AP_FIRST_IN_LIST.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

buf_size
提供されたバッファーに戻された情報の長さ。

total_buf_size
要求されたすべてのリスト情報を戻すために必要となるバッファーのサイズを示す戻り値。 *buf_size* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

num_entries
データ・バッファーに戻されたエントリーの数。

total_num_entries
使用可能な項目の合計数。 *num_entries* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

データ・バッファー内の各項目は、以下のパラメーターで構成されます。

ローカル・トポロジー・サマリー . オーバー・サイズ
戻された local_topology_summary 構造体のサイズ。すなわち、データ・バッファー内の次のエントリーの先頭までのオフセット。

アプリケーションが戻されたバッファーを調べて、各 local_topology_summary 構造体を順番に検出する必要がある場合は、この値を使用して次のデータ構造の正しいオフセットに移動しなければなりません。また、 C sizeof() 演算子を使用してはなりません。 これは、返されるオーバーレイのサイズが、 CS Linux の将来のリリースで増加する可能性があるためです。戻されたオーバーレイ・サイズを使用すると、アプリケーションが今後のリリースで引き続き動作することが保証されます。

ローカル・トポロジー要約 . デスト
TG の完全修飾宛先ノード名。 この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、 1 から 8 文字の A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および 1 から 8 文字の A ストリング文字のネットワーク名で構成されます。

ローカル・トポロジーの要約 . *dest_type*
この TG の宛先ノードのノード・タイプ。 これは、以下のいずれかです。

AP_NETWORK_NODE
ネットワーク・ノード。

ファイルの追加
仮想ルーティング・ノード。

ノードの追加 (_L)
エンド・ノードまたは LEN ノード。

ローカル・トポロジーの要約 *. tg_num*
TG に関連付けられている番号。

ローカル・トポロジー詳細 . オーバーレイ・サイズ
戻された local_topology_detail 構造体のサイズ。すなわち、データ・バッファー内の次のエントリーの先頭までのオフセット。

アプリケーションが戻されたバッファーを調べて、各 local_topology_detail 構造体を順番に検出する必要がある場合は、この値を使用して次のデータ構造の正しいオフセットに移動しなければなりません。また、 C sizeof() 演算子を使用してはなりません。 これは、返されるオーバーレイのサイズが、 CS Linux の将来のリリースで増加する可能性があるためです。戻されたオーバーレイ・サイズを使用すると、アプリケーションが今後のリリースで引き続き動作することが保証されます。

ローカル・トポロジー詳細 *.dest*
TG の完全修飾宛先ノード名。 この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、 1 から 8 文字の A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および 1 から 8 文字の A ストリング文字のネットワーク名で構成されます。

ローカル・トポロジー詳細 *. dest_type*
この TG の宛先ノードのノード・タイプ。 これは、以下のいずれかです。

AP_NETWORK_NODE
ネットワーク・ノード。

ファイルの追加
仮想ルーティング・ノード。

ノードの追加 (_L)
エンド・ノードまたは LEN ノード。

ローカル・トポロジーの詳細 *. tg_num*
TG に関連付けられている番号。

ローカル・トポロジーの詳細 *.dlc_data.length*
dest_type が **ファイルの追加**の場合、このフィールドは VRN への接続の DLC アドレスの長さを指定します。 それ以外の場合、このフィールドは使用されず、ゼロに設定されます

ローカル・トポロジーの詳細 *.dlc_data.address*
dest_type が **ファイルの追加**の場合、このフィールドは VRN への接続の DLC アドレス (16 進数) を指定します。 アドレス内のバイト数は、直前のフィールドの長さによって与えられます。フィールドの残りのバイト数は未定義です。 それ以外の場合、このフィールドは使用されませ

トークンリングまたはイーサネットの場合、アドレスは、 6 バイトの MAC アドレスと 1 バイトのローカル SAP アドレスの 2 つの部分にあります。 MAC アドレスのビット配列が、予期されるフォーマットになっていない可能性があります。 2 つのアドレス・フォーマットの間の変換については、 124 ページの『Bit ordering in MAC addresses』を参照してください。

ローカル・トポロジー詳細 *.rsn*
リソース・シーケンス番号。 これは、このリソースを所有するネットワーク・ノードによって割り当てられます。

ローカル・トポロジー詳細 . 状況
TG の状況を指定します。 これは、論理 **それとも** 操作によって結合された、以下の 1 つ以上の可能性があります。

作動可能の状態

TG_CP_CP_SESSIONS セッション

_TG_静止中

追加指定の HP_HPR

追加要求の要求

ローカル・トポロジー詳細 . *tg_chars*
TG 特性。 これらのパラメーターについて詳しくは、 104 ページの『DEFINE_LS』を参照してください。

local_topology_detail.cp_cp_session_active
所有ノードの競合勝者 CP-CP セッションがアクティブであるかどうかを指定します。 可能な値は次のとおりです

類人猿
CP-CP セッションは活動状態です。

アプ・ノー
CP-CP セッションは活動状態ではありません。

不明
CP-CP セッション状況は不明です。

ローカル・トポロジーの詳細 . *branch_link_type*
このパラメーターは、ノードが分岐ネットワーク・ノードである場合にのみ適用されます。それ以外の場合は予約済みです

この TG のブランチ・リンク・タイプを指定します。 可能な値は次のとおりです

アプ・アップリンク
TG はアップリンクです。

AP_DOWNLINK
TG は、エンド・ノードへのダウンリンクです。

リンク・リンクの追加 （ _BRNN ）
TG は、ローカル・ノードの観点から、エンド・ノードとして表示される分岐ネットワーク・ノードへのダウンリンクです。

アプアザリンク
TG は VRN へのリンクです。

local_topology_detail.branch_tg
このパラメーターは、ノードがネットワーク・ノードである場合にのみ適用されます。それ以外の場合は予約済み

TG が分岐 TG であるかどうかを指定します。 可能な値は次のとおりです

類人猿
TG は分岐 TG です。

アプ・ノー
TG は分岐 TG ではありません。

不明
TG タイプは不明です。

ローカル・トポロジー詳細 . *appended_data_format*
この NOF VCB 構造体に付加されるデータのフォーマットを指定します。

パラメーター ローカル・トポロジー詳細 . *appended_data_len* がゼロ以外の値に設定されている場合、付加されたデータが含まれていることを示す場合、このパラメーターは以下の値に設定されます。

ターゲット・ディスクリプター CV
付加されたデータには、 SNA 形式で定義されているように、 TG 記述子 CV が含まれます。

ローカル・トポロジー詳細 . *appended_data_len* がゼロの場合、付加されたデータが含まれていないことを示すパラメーターは予約済みです。

ローカル・トポロジー詳細 . *appended_data_len*
この NOF VCB 構造体に付加される TG 記述子 CV データの長さを指定します。 このパラメーターがゼロに設定されている場合、追加されたデータは組み込まれません。

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_TG
The *list_options* parameter was set to AP_LIST_INCLUSIVE to list all entries starting from the supplied name, but the *tg_num* parameter was not valid.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

照会ログ・ファイル

この verb を使用すると、アプリケーションは、 CS Linux が監査、エラー、または使用量のログ・メッセージを記録するために使用するファイルの名前、バックアップ・ログ・ファイルの名前、およびログ情報がバックアップ・ファイルにコピーされるファイル・サイズを決定することができます。

VCB 構造体

```
typedef struct query_log_file
{
   AP_UINT16         opcode;                  /* verb operation code     */
   unsigned char     reserv2;                 /* reserved                */
   unsigned char     format;                  /* reserved                */
   AP_UINT16         primary_rc;              /* primary return code     */
   AP_UINT32         secondary_rc;            /* secondary return code   */
   unsigned char     log_file_type;           /* type of log file        */
   unsigned char     file_name[81];           /* file name               */
   unsigned char     backup_file_name[81];    /* backup file name        */
   AP_UINT32         file_size;               /* log file size           */
   unsigned char     succinct;                /* reserved                */
   unsigned char     reserv3[3];              /* reserved                */
} QUERY_LOG_FILE;
```

提供されるパラメーター

オペコード
アプリケーション・ログ・ファイル

ログ・ファイル・タイプ
照会されるログ・ファイルのタイプ。 可能な値は次のとおりです

ファイルの追加ファイル
監査ログ・ファイル (監査メッセージのみ)。

AP_ERROR_FILE
エラー・ログ・ファイル (問題メッセージおよび例外メッセージ)。

AP_USAGE_FILE
使用量ログ・ファイル (CS Linux リソースの現在およびピーク使用量に関する情報) 。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

secondary_rc
未使用。

ファイル名
ログ・ファイルの名前。 このパラメーターは、 1 文字から 80 文字の ASCII ストリングで、その後にヌル (0x00) 文字が続きます。

パスが含まれていない場合、ファイルは診断ファイル /var/opt/ibm/sna。パスが含まれている場合、これは、絶対パス (/ 文字で始まる) またはデフォルト・ディレクトリーに対する相対パスのいずれかになります。 のデフォルト・ディレクトリーに保管されます。

バックアップ・ファイル名
バックアップ・ログ・ファイルの名前。 このパラメーターは、 1 文字から 80 文字の ASCII ストリングで、その後にヌル (0x00) 文字が続きます。

以下の ファイル・サイズ で指定されたサイズにログ・ファイルが達すると、 CS Linux はログ・ファイルの現在の内容をこのファイルにコピーしてから、ログ・ファイルをクリアします。 また、 SET_LOG_FILE verb を使用して、いつでもバックアップを要求することができます。

パスが含まれていない場合、ファイルは診断ファイル /var/opt/ibm/sna。パスが含まれている場合、これは、絶対パス (/ 文字で始まる) またはデフォルト・ディレクトリーに対する相対パスのいずれかになります。 のデフォルト・ディレクトリーに保管されます。

ファイル・サイズ
ログ・ファイル・タイプによって指定されるログ・ファイルの最大サイズ。 ファイルに書き込まれたメッセージによってファイル・サイズがこの制限を超えると、 CS Linux はバックアップ・ログ・ファイルをクリアし、ログ・ファイルの現在の内容をバックアップ・ログ・ファイルにコピーしてから、ログ・ファイルをクリアします。 これは、ログ・ファイルによって取られるディスク・スペースの最大量が、 ファイル・サイズの値の約 2 倍になることを意味します。

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc

AP_INVALID_FILE_TYPE
The *log_file_type* parameter was not set to a valid value.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

QUERY_LOG_TYPE

This verb allows a NOF application to determine the types of information that CS Linux records in log files on a particular server, and whether these are the default settings specified on SET_GLOBAL_LOG_TYPE or local settings specified by a previous SET_LOG_TYPE verb.

CS Linux logs messages for the following types of event:

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

Exception
An abnormal event that may degrade the system but that is not immediately perceptible to a user (such as receiving a message that is not valid from the remote system).

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

Problem and exception messages are logged to the error log file; audit messages are logged to the audit log file. Problem messages are always logged and cannot be disabled, but you can specify whether to log each of the other two message types. For each of the two files (audit and error), you can specify whether to use succinct logging (including only the text of the message and a summary of the message source) or full logging (including full details of the message source, cause, and any action required).

VCB 構造体

```
typedef struct query_log_type
{
   AP_UINT16       opcode;            /* verb operation code                */
   unsigned char   reserv2;           /* reserved                           */
   unsigned char   format;            /* reserved                           */
   AP_UINT16       primary_rc;        /* primary return code                */
   AP_UINT32       secondary_rc;      /* secondary return code              */
   unsigned char   override;          /* overriding global settings?        */
   unsigned char   audit;             /* audit logging on or off            */
   unsigned char   exception;         /* exception logging on or off        */
   unsigned char   succinct_audits;   /* use succinct logging in audit file? */
   unsigned char   succinct_errors;   /* use succinct logging in error file? */
   unsigned char   reserv3[3];        /* reserved                           */
} QUERY_LOG_TYPE;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
アプリケーション・ログ・タイプの追加

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

secondary_rc
Not used.

override
Specifies whether the log types and succinct or full logging options returned on this verb are the global log types specified on SET_GLOBAL_LOG_TYPE, or local values specified on SET_LOG_TYPE. Possible values are:

AP_YES
The *audit*, *exception*, and *succinct_** parameters returned are local settings overriding the global settings.

AP_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:

AP_YES
Audit messages are recorded.

AP_NO
Audit messages are not recorded.

exception
This parameter indicates whether exception messages are recorded. Possible values are:

AP_YES
Exception messages are recorded.

AP_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:

AP_YES
Succinct logging: 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 of the cause of the log and any action required, you can use the `snahelp` utility.

AP_NO
Full logging: each message in the log file includes a full listing of the message header information, the message text string and parameters, and additional information about the cause of the log and any action required.

If you are using central logging, the choice of succinct or full logging for messages from all computers is determined by the setting of this parameter on the server acting as the central logger; this setting may either be from the SET_GLOBAL_LOG_TYPE verb, or from a SET_LOG_TYPE verb issued to that server to override the default.

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. The possible values and their meanings are the same as for the *succinct_audits* parameter.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

QUERY_LS

QUERY_LS returns a list of information about the link stations defined at the node. This information is structured as "determined data" (data gathered dynamically during execution, returned only if the node is active) and "defined data" (data supplied on DEFINE_LS).

This verb can be used to obtain either summary or detailed information, about a specific LS or about multiple LSs, depending on the options used.

VCB 構造体

```
typedef struct query_ls
{
   AP_UINT16        opcode;                 /* verb operation code          */
   unsigned char    reserv2;                /* reserved                     */
   unsigned char    format;                 /* reserved                     */
   AP_UINT16        primary_rc;             /* Primary return code          */
   AP_UINT32        secondary_rc;           /* Secondary return code        */
   unsigned char    *buf_ptr;               /* pointer to buffer            */
   AP_UINT32        buf_size;               /* buffer size                  */
   AP_UINT32        total_buf_size;         /* total buffer size required   */
```

```
   AP_UINT16        num_entries;            /* number of entries            */
   AP_UINT16        total_num_entries;      /* total number of entries      */
   unsigned char    list_options;           /* listing options              */
   unsigned char    reserv3;                /* reserved                     */
   unsigned char    ls_name[8];             /* name of link station         */
   unsigned char    port_name[8];           /* port used by link station    */
} QUERY_LS;
```

```
typedef struct ls_summary
{
   AP_UINT16        overlay_size;           /* size of returned entry       */
   unsigned char    ls_name[8];             /* link station name            */
   unsigned char    description[32];        /* resource description         */
   unsigned char    reserv1[16];            /* reserved                     */
   unsigned char    dlc_type;               /* DLC type                     */
   unsigned char    state;                  /* link station state           */
   AP_UINT16        act_sess_count;         /* currently active sessions    */
                                            /*  count                       */
   unsigned char    det_adj_cp_name[17];    /* determined adjacent CP name  */
   unsigned char    det_adj_cp_type;        /* determined adjacent node type*/
   unsigned char    port_name[8];           /* port name                    */
   unsigned char    adj_cp_name[17];        /* adjacent CP name             */
   unsigned char    adj_cp_type;            /* adjacent node type           */
} LS_SUMMARY;
```

```
typedef struct ls_detail
{
   AP_UINT16        overlay_size;           /* size of returned entry       */
   unsigned char    ls_name[8];             /* link station name            */
   LS_DET_DATA      det_data;               /* determined data              */
   LS_DEF_DATA      def_data;               /* defined data                 */
} LS_DETAIL;
```

```
typedef struct ls_det_data
{
   AP_UINT16        act_sess_count;         /* currently active sessions    */
                                            /*  count                       */
   unsigned char    dlc_type;               /* DLC type                     */
   unsigned char    state;                  /* link station state           */
   unsigned char    sub_state;              /* link station sub state       */
   unsigned char    det_adj_cp_name[17];    /* adjacent CP name             */
   unsigned char    det_adj_cp_type;        /* adjacent node type           */
   unsigned char    dlc_name[8];            /* name of DLC                  */
   unsigned char    dynamic;                /* specifies whether LS is      */
                                            /*  dynamic                     */
   unsigned char    migration;              /* supports migration partners  */
   unsigned char    tg_num;                 /* TG number                    */
   LS_STATS         ls_stats;               /* link station statistics      */
   AP_UINT32        start_time;             /* time LS started              */
   AP_UINT32        stop_time;              /* time LS stopped              */
   AP_UINT32        up_time;                /* total time LS active         */
   AP_UINT32        current_state_time;     /* time in current state        */
   unsigned char    deact_cause;            /* deactivation cause           */
   unsigned char    hpr_support;            /* TG HPR support               */
   unsigned char    anr_label[2];           /* local ANR label              */
   unsigned char    hpr_link_lvl_error;     /* HPR link-level error         */
   unsigned char    auto_act;               /* auto-activation supported    */
   unsigned char    ls_role;                /* LS role                      */
   unsigned char    ls_type;                /* LS type (defined,dynamic,..) */
   unsigned char    node_id[4];             /* determined node ID           */
   AP_UINT16        active_isr_count;       /* active isr count             */
   AP_UINT16        active_lu_sess_count;   /* count of active LU sessions  */
   AP_UINT16        active_sscp_sess_count; /* count of active SSCP sessions*/
   ANR_LABEL        reverse_anr_label;      /* Reverse ANR label            */
   LINK_ADDRESS     local_address;          /* Local address                */
   AP_UINT16        max_send_btu_size;      /* Max send BTU size            */
   unsigned char    brnn_link_type;         /* type of branch link          */
   unsigned char    adj_cp_is_brnn;         /* is adjacent node a BrNN?     */
   unsigned char    mltg_member;            /* reserved                     */
   unsigned char    tg_sharing;             /* reserved                     */
   unsigned char    reservb[62];            /* reserved                     */
} LS_DET_DATA;
```

```
typedef struct ls_def_data
{
   unsigned char    description[32];        /* resource description         */
   unsigned char    initially_active;       /* is this LS initially active? */
   unsigned char    reserv2;                /* reserved                     */
```

```
   AP_UINT16        react_timer;             /* timer for retrying failed LS */
   AP_UINT16        react_timer_retry;       /* retry count for failed LS    */
   AP_UINT16        activation_count;        /* reserved                     */
   unsigned char    restart_on_normal_deact; /* restart the link on any      */
                                             /* failure                      */
   unsigned char    reserv3[7];              /* reserved                     */
   unsigned char    port_name[8];            /* name of associated port      */
   unsigned char    adj_cp_name[17];         /* adjacent CP name             */
   unsigned char    adj_cp_type;             /* adjacent node type           */
   LINK_ADDRESS     dest_address;            /* destination address          */
   unsigned char    auto_act_supp;           /* auto-activate supported      */
   unsigned char    tg_number;               /* pre-assigned TG number       */
   unsigned char    limited_resource;        /* limited resource             */
   unsigned char    solicit_sscp_sessions;   /* solicit SSCP sessions        */
   unsigned char    pu_name[8];              /* Local PU name (reserved if   */
                                             /*  solicit_sscp_sessions is    */
                                             /*  set to AP_NO)               */
   unsigned char    disable_remote_act;      /* disable remote activation    */
   unsigned char    dspu_services;           /* Services provided for        */
                                             /*  downstream PU               */
   unsigned char    dspu_name[8];            /* Downstream PU name (reserved */
                                             /*  if dspu_services is AP_NONE)*/
   unsigned char    dlus_name[17];           /* DLUS name if dspu_services   */
                                             /*  is AP_DLUR                  */
   unsigned char    bkup_dlus_name[17];      /* Backup DLUS name if          */
                                             /*  dspu_services is AP_DLUR    */
   unsigned char    hpr_supported;           /* does the link support HPR?   */
   unsigned char    hpr_link_lvl_error;      /* does link use link-level     */
                                             /*  recovery for HPR frames?    */
   AP_UINT16        link_deact_timer;        /* deact timer for limited      */
                                             /*  resource                    */
   unsigned char    reserv1;                 /* reserved                     */
   unsigned char    default_nn_server;       /* default LS to NN server?     */
   unsigned char    ls_attributes[4];        /* LS attributes                */
   unsigned char    adj_node_id[4];          /* adjacent node ID             */
   unsigned char    local_node_id[4];        /* local node ID                */
   unsigned char    cp_cp_sess_support;      /* CP-CP session support        */
   unsigned char    use_default_tg_chars;    /* Use default tg_chars         */
   TG_DEFINED_CHARS tg_chars;                /* TG characteristics           */
   AP_UINT16        target_pacing_count;     /* target pacing count          */
   AP_UINT16        max_send_btu_size;       /* maximum send BTU size        */
   AP_UINT16        ls_role;                 /* link station role            */
   unsigned char    max_ifrm_rcvd;           /* no. before acknowledgment    */
   AP_UINT16        dlus_retry_timeout;      /* seconds to recontact a DLUS  */
   AP_UINT16        dlus_retry_limit;        /* attempts to recontact a DLUS */
   unsigned char   conventional_lu_compression; /* compression for LU 0-3?   */
   unsigned char    conventional_lu_cryptography; /* reserved               */
   unsigned char    reserv3a;                 /* reserved                    */
   unsigned char    retry_flags;              /* reserved                    */
   AP_UINT16        max_activation_attempts;  /* reserved                    */
   AP_UINT16        activation_delay_timer;   /* reserved                    */
   unsigned char    branch_link_type;         /* is link an up or down link  */
   unsigned char    adj_brnn_cp_support;      /* adj CP allowed to be BrNN?  */
   unsigned char    mltg_pacing_algorithm;    /* reserved                    */
   unsigned char    reserv5;                  /* reserved                    */
   AP_UINT16        max_rcv_btu_size;         /* reserved                    */
   unsigned char    tg_sharing_prohibited;    /* reserved                    */
   unsigned char    link_spec_data_format;    /* reserved                    */
   unsigned char    pu_can_send_dddlu_offline;  /* does the PU send NMVT     */
                                                /*  (power off) to the host? */
   unsigned char    reserv4[13];              /* reserved                    */
   AP_UINT16        link_spec_data_len;       /* length of link specific data */
} LS_DEF_DATA;
```

```
typedef struct link_address
{
  unsigned char   format;                 /* type of link address          */
  unsigned char   reserve1;               /* reserved                      */
  AP_UINT16       length;                 /* length                        */
  unsigned char   address[32];            /* address                       */
} LINK_ADDRESS;
```

トークンリングまたはイーサネットの場合、 **リンク・アドレス** 構造体内の アドレス パラメーターは、以下のように置き換えられます。

```
typedef struct tr_address
{
   unsigned char    mac_address[6];          /* MAC address                 */
```

```
   unsigned char    lsap_address;            /* local SAP address           */
} TR_ADDRESS;
```

Enterprise Extender (HPR/IP) の場合、 **リンク・アドレス** 構造体内の アドレス パラメーターは、以下のように置き換えられます。

```
typedef struct ip_address_info
{
  unsigned char lsap;                      /* Local Service Access Point addr */
  unsigned char version;                   /* IPv4 or IPv6                    */
  unsigned char address[272];              /* IP Address or hostname          */

} IP_ADDRESS_INFO;
```

マルチパス・チャネル (MPC) または MPC+の場合、 **リンク・アドレス** 構造体内の アドレス パラメーターは、以下のように置き換えられます。

```
typedef unsigned char GDLC_MPC_ADDRESS[20];
```

すべてのリンク・タイプ:

```
typedef struct tg_defined_chars
{
   unsigned char    effect_cap;              /* Effective capacity          */
   unsigned char    reserve1[5];             /* Reserved                    */
   unsigned char    connect_cost;            /* Connection Cost             */
   unsigned char    byte_cost;               /* Byte cost                   */
   unsigned char    reserve2;                /* Reserved                    */
   unsigned char    security;                /* Security                    */
   unsigned char    prop_delay;              /* Propagation delay           */
   unsigned char    modem_class;             /* reserved                    */
   unsigned char    user_def_parm_1;         /* User-defined parameter 1    */
   unsigned char    user_def_parm_2;         /* User-defined parameter 2    */
   unsigned char    user_def_parm_3;         /* User-defined parameter 3    */
} TG_DEFINED_CHARS;
```

```
typedef struct ls_stats
{
   AP_UINT32       in_xid_bytes;         /* number of XID bytes received     */
   AP_UINT32       in_msg_bytes;         /* number of message bytes received */
   AP_UINT32       in_xid_frames;        /* number of XID frames received    */
   AP_UINT32       in_msg_frames;        /* number of message frames received*/
   AP_UINT32       out_xid_bytes;        /* number of XID bytes sent         */
   AP_UINT32       out_msg_bytes;        /* number of message bytes sent     */
   AP_UINT32       out_xid_frames;       /* number of XID frames sent        */
   AP_UINT32       out_msg_frames;       /* number of message frames sent    */
   AP_UINT32       in_invalid_sna_frames;       /* number of invalid frames  */
                                        /*  received                        */
   AP_UINT32       in_session_control_frames;   /* number of control frames  */
                                        /*  received                        */
   AP_UINT32       out_session_control_frames;  /* number of control frames  */
                                        /*  sent                            */
   AP_UINT32       echo_rsps;           /* reserved                         */
   AP_UINT32       current_delay;       /* reserved                         */
   AP_UINT32       max_delay;           /* reserved                         */
   AP_UINT32       min_delay;           /* reserved                         */
   AP_UINT32       max_delay_time;      /* reserved                         */
   AP_UINT32       good_xids;           /* successful XID on LS count       */
   AP_UINT32       bad_xids;            /* unsuccessful XID on LS count     */
} LS_STATS;
```

リンク固有データの詳細については、 104 ページの『DEFINE_LS』を参照してください。 このデータのデータ構造は、 `ls_def_data` 構造体の後に続きますが、 4 バイトの境界で開始するように埋め込みが行われます。

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_LS

buf_ptr
A pointer to a data buffer that CS Linux will use to return the requested information.

buf_size
Size of the supplied data buffer.

num_entries
Maximum number of LSs for which data should be returned. To request data for a specific LS rather than a range, specify the value 1. To return as many entries as possible, specify zero; in this case, CS Linux will return the maximum number of entries that can be accommodated in the supplied data buffer.

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

AP_SUMMARY
Summary information only.

AP_DETAIL
Detailed information.

Combine this value using a logical OR operation with one of the following values:

AP_FIRST_IN_LIST
Start at the first entry in the list.

AP_LIST_INCLUSIVE
Start at the entry specified by the *ls_name* parameter.

AP_LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *ls_name* parameter.

For more information about how the list is ordered and how the application can obtain specific entries from it, see "List options for QUERY_* Verbs" on page 34.

ls_name
Link station name. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 characters. This value is ignored if *list_options* is set to AP_FIRST_IN_LIST.

port_name
Port name filter. To return information only on LSs associated with a specific port, specify the name of the port. This is an 8-byte ASCII string), padded on the right with spaces if the name is shorter than 8 characters. To return information about all LSs without filtering on the port name, set this parameter to 8 binary zeros.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

ls_summary.overlay_size
The size of the returned `ls_summary` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `ls_summary` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

ls_summary.ls_name
Link station name. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 characters.

ls_summary.description
A null-terminated text string describing the LS, as specified in the definition of the LS.

ls_summary.dlc_type
Type of DLC. This is one of the following:

AP_SDLC
SDLC

AP_X25
QLLC

AP_TR
Token Ring

AP_ETHERNET
Ethernet

AP_MPC
Multipath Channel (MPC), CS Linux for IBM Z only

AP_IP
Enterprise Extender (HPR/IP)

ls_summary.state
State of this link station. This is one of the following:

AP_ACTIVE
The LS is active.

AP_NOT_ACTIVE
The LS is not active.

AP_PENDING_ACTIVE
The LS is being activated.

AP_PENDING_INACTIVE
The LS is being deactivated.

AP_PENDING_ACTIVE_BY_LR
The LS has failed (or an attempt to activate it has failed), and CS Linux is attempting to reactivate it.

ls_summary.act_sess_count
The total number of active sessions (both endpoint and intermediate) using the link.

ls_summary.det_adj_cp_name
Fully qualified name of the adjacent control point. The name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of 1-8 A-string characters, an EBCDIC dot (period) character, and a network name of 1-8 A-string characters.

This name is normally 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.

ls_summary.det_adj_cp_type
Type of the adjacent node. This is one of the following:

AP_APPN_NODE
Node type is unknown, or LS is inactive.

AP_END_NODE
End node, Branch Network Node acting as an End Node from the local node's perspective, or up-level LEN node (one that includes the Network Name CV in its XID3).

AP_NETWORK_NODE
Network node, or Branch Network Node acting as a Network Node from the local node's perspective.

AP_VRN
Virtual routing node.

The node type is normally 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.

ls_summary.port_name
Name of the port associated with this link station. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 characters.

ls_summary.adj_cp_name
Fully qualified name of the adjacent control point; this parameter is null for an implicit link. The name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of 1-8 A-string characters, an EBCDIC dot (period) character, and a network name of 1-8 A-string characters.

ls_summary.adj_cp_type
Type of the adjacent node, determined during link activation. This is one of the following:

AP_APPN_NODE
Node type is unknown, or LS is inactive.

AP_END_NODE
End node, Branch Network Node acting as an End Node from the local node's perspective, or up-level LEN node (one that includes the Network Name CV in its XID3).

AP_NETWORK_NODE
Network node, or Branch Network Node acting as a Network Node from the local node's perspective.

AP_BACK_LEVEL_LEN_NODE
Back-level LEN node (one that does not include the Network Name CV in its XID3).

AP_HOST_XID3
Host node; CS Linux should respond to a polling XID from the node with a format 3 XID.

AP_HOST_XID0
Host node; CS Linux should respond to a polling XID from the node with a format 0 XID.

AP_DSPU_XID
Downstream PU; CS Linux should include XID exchange in link activation. The *dspu_name* and *dspu_services* fields must also be set.

AP_DSPU_NOXID
Downstream PU; CS Linux should not include XID exchange in link activation. The *dspu_name* and *dspu_services* fields must also be set.

AP_VRN
Virtual routing node.

ls_detail.overlay_size
The size of the returned `ls_detail` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `ls_detail` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

ls_detail.ls_name
Link station name. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 characters.

ls_detail.det_data.act_sess_count
The total number of active sessions (both endpoint and intermediate) using the link.

ls_detail.det_data.dlc_type
Type of DLC. This is one of the following:

AP_SDLC
SDLC

AP_X25
QLLC

AP_TR
Token Ring

AP_ETHERNET
Ethernet

AP_MPC
Multipath Channel (MPC), CS Linux for IBM Z only

AP_IP
Enterprise Extender (HPR/IP)

ls_detail.det_data.state
State of this link station. This is one of the following:

AP_ACTIVE
The LS is active.

AP_NOT_ACTIVE
The LS is not active.

AP_PENDING_ACTIVE
The LS is being activated.

AP_PENDING_INACTIVE
The LS is being deactivated.

AP_PENDING_ACTIVE_BY_LR
The LS has failed (or an attempt to activate it has failed), and CS Linux is attempting to reactivate it.

ls_detail.det_data.sub_state
This field provides more detailed information about the state of this link station. Possible values are:

`AP_SENT_CONNECT_OUT`

`AP_PENDING_XID_EXCHANGE`

`AP_SENT_ACTIVATE_AS`

`AP_SENT_SET_MODE`

`AP_ACTIVE`

`AP_SENT_DEACTIVATE_AS_ORDERLY`

`AP_SENT_DISCONNECT`

`AP_WAITING_STATS`

AP_RESET

ls_detail.det_data.det_adj_cp_name
Fully qualified name of the adjacent control point. The name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of 1-8 A-string characters, an EBCDIC dot (period) character, and a network name of 1-8 A-string characters.

This name is normally 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.

ls_detail.det_data.det_adj_cp_type
Type of the adjacent node. This is one of the following:

AP_END_NODE
End node, or Branch Network Node acting as an End Node from the local node's perspective.

AP_NETWORK_NODE
Network node, or Branch Network Node acting as a Network Node from the local node's perspective.

AP_LEARN_NODE
Node type is unknown.

The node type is normally 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.

ls_detail.det_data.dlc_name
Name of the DLC. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 characters.

ls_detail.det_data.dynamic
Specifies whether the link was defined dynamically. Possible values are:

AP_YES
The link was defined dynamically (in response to a connection request from the adjacent node, or to connect dynamically to another node across a Connection Network).

AP_NO
The link was defined explicitly (by DEFINE_LS).

ls_detail.det_data.migration
Specifies whether the adjacent node is a migration level node (such as a Low Entry Networking or LEN node), or a full APPN network node or end node. Possible values are:

AP_YES
The adjacent node is a migration-level node.

AP_NO
The adjacent node is a network node or end node.

AP_UNKNOWN
The adjacent node level is unknown.

ls_detail.det_data.tg_num
Number associated with the TG.

ls_detail.det_data.ls_stats.in_xid_bytes
Total number of XID (Exchange Identification) bytes received on this link station.

ls_detail.det_data.ls_stats.in_msg_bytes
Total number of data bytes received on this link station.

ls_detail.det_data.ls_stats.in_xid_frames
Total number of XID (Exchange Identification) frames received on this link station.

ls_detail.det_data.ls_stats.in_msg_frames
Total number of data frames received on this link station.

ls_detail.det_data.ls_stats.out_xid_bytes
Total number of XID (Exchange Identification) bytes sent on this link station.

ls_detail.det_data.ls_stats.out_msg_bytes
Total number of data bytes sent on this link station.

ls_detail.det_data.ls_stats.out_xid_frames
Total number of XID (Exchange Identification) frames sent on this link station.

ls_detail.det_data.ls_stats.out_msg_frames
Total number of data frames sent on this link station.

ls_detail.det_data.ls_stats.in_invalid_sna_frames
Total number of not valid SNA frames received on this link station.

ls_detail.det_data.ls_stats.in_session_control_frames
Total number of session control frames received on this link station.

ls_detail.det_data.ls_stats.out_session_control_frames
Total number of session control frames sent on this link station.

ls_detail.det_data.ls_stats.good_xids
Total number of successful XID exchanges that have occurred on this link station since it was started.

ls_detail.det_data.ls_stats.bad_xids
Total number of unsuccessful XID exchanges that have occurred on this link station since it was started.

ls_detail.det_data.start_time
Time since system startup (in hundredths of a second) when the link station was last activated (that is, when the mode setting commands completed).

ls_detail.det_data.stop_time
Time since system startup (in hundredths of a second) when the link station was last deactivated.

ls_detail.det_data.up_time
Total time (in hundredths of a second) that this link station has been active since system startup.

ls_detail.det_data.current_state_time
Total time (in hundredths of a second) that this link station has been in its current state.

ls_detail.det_data.deact_cause
The cause of the last deactivation of the link station. Possible values are:

AP_NONE
The link station has never been deactivated.

AP_DEACT_OPER_ORDERLY
The link station was deactivated as a result of an orderly STOP command from an operator.

AP_DEACT_OPER_IMMEDIATE
The link station was deactivated as a result of an immediate STOP command from an operator.

AP_DEACT_AUTOMATIC
The link station was deactivated automatically, for example because there were no more sessions using the link station.

AP_DEACT_FAILURE
The link station was deactivated because of a failure.

ls_detail.det_data.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:

AP_NONE
This TG does not support HPR protocols.

AP_BASE
This TG supports base level HPR.

AP_RTP
This TG supports Rapid Transport Protocols (RTP).

ls_detail.det_data.anr_label
The HPR automatic network routing (ANR) label allocated to the local link.

ls_detail.det_data.hpr_link_lvl_error
For SDLC, this parameter is reserved.

For other port types, this parameter specifies whether link-level error recovery is being used for HPR traffic on the link.

ls_detail.det_data.auto_act
Specifies whether the link currently allows remote activation or activation on demand. This is set to AP_NONE if neither is allowed, or to one or both of the following values (combined using a logical OR):

AP_AUTO_ACT
The link can be activated on demand by the local node when a session requires it.

AP_REMOTE_ACT
The link can be activated by the remote node.

ls_detail.det_data.ls_role
The LS role of this link. This is normally 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 to be negotiable, it will be negotiated to either primary or secondary while the LS is active; if this verb is used to query an active LS, the returned LS role is the negotiated role currently in use and not the defined role. Possible values are:

AP_LS_PRI
Primary.

AP_LS_SEC
Secondary.

AP_LS_NEG
Negotiable.

ls_detail.det_data.ls_type
Specifies how this link was defined or discovered. Possible values are:

AP_LS_DEFINED
The link station was defined explicitly by a CS Linux administration program.

AP_LS_DYNAMIC
The link station was created when the local node connected to another node through a connection network.

AP_LS_TEMPORARY
The link station was created temporarily to process an incoming call, but has not yet become active.

AP_LS_IMPLICIT
The link station was defined implicitly when CS Linux received an incoming call that it could not match to a defined link station.

AP_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.

ls_detail.det_data.node_id
Node ID received from adjacent node during XID exchange. This is a 4-byte hexadecimal string.

ls_detail.det_data.active_isr_count
Number of active intermediate sessions using the link.

ls_detail.det_data.active_lu_sess_count
The count of active LU-LU sessions using this link.

ls_detail.det_data.active_sscp_sess_count
The count of active PU-SSCP sessions using this link.

ls_detail.det_data.reverse_anr_label
The Reverse Automatic Network Routing (ANR) label for this link station.

For SDLC:

ls_detail.det_data.local_address
The local address of this link station.

For QLLC:

ls_detail.det_data.local_address
The local address of this link station.

For Token Ring, Ethernet:

ls_detail.det_data.local_address.mac_address
MAC address of the local link station.

ls_detail.det_data.local_address.lsap_address
Local SAP address of the local link station.

For Enterprise Extender:

ls_detail.det_data.local_address.ip_address_info.lsap
For Enterprise Extender: Local SAP address of the port. Specify a multiple of `0x04` in the range `0x04`-`0xEC`. The usual value is `0x04`, but VTAM may use `0x08` 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.

ls_detail.det_data.local_address.ip_address_info.version
For Enterprise Extender: Specifies whether the following field represents an IPv4 or IPv6 address. Possible values:

IP_VERSION_4_HOSTNAME
The *address* field specifies an IPv4 address, or a hostname or alias that resolves to an IPv4 address.

IP_VERSION_6_HOSTNAME
The *address* field specifies an IPv6 address, or a hostname or alias that resolves to an IPv6 address.

ls_detail.det_data.local_address.ip_address_info.address
For Enterprise Extender: IP address of the port. 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`).

For multipath channel (MPC), CS Linux for IBM Z only:

ls_detail.det_data.local_address.address
This parameter is reserved.

ls_detail.det_data.max_send_btu_size
Maximum BTU size that can be sent on this link, as determined by negotiation with the adjacent node. If the link activation has not yet been attempted, a zero value is returned.

ls_detail.det_data.brnn_link_type
This parameter applies only if the local node is a Branch Network Node; it is reserved otherwise.

Specifies the branch link type of this link. Possible values are:

AP_UPLINK
The link is an uplink.

AP_DOWNLINK
The link is a downlink.

AP_OTHERLINK
The link is to a VRN.

AP_UNKNOWN_LINK_TYPE
The branch link type is unknown.

AP_BRNN_NOT_SUPPORTED
The link supports PU 2.0 traffic only.

ls_detail.det_data.adj_cp_is_brnn
Specifies whether the adjacent node is a Branch Network Node. Possible values are:

AP_YES
The adjacent node is a Branch Network Node.

AP_NO
The adjacent node is not a Branch Network Node.

AP_UNKNOWN
The adjacent node type is unknown.

ls_detail.def_data.description
A null-terminated text string describing the LS, as specified in the definition of the LS.

ls_detail.def_data.initially_active
Specifies whether this LS is automatically started when the node is started. Possible values are:

AP_YES
The LS is automatically started when the node is started.

AP_NO
The LS is not automatically started; it must be started manually.

ls_detail.def_data.react_timer
Reactivation timer for reactivating a failed LS. If the *react_timer_retry* parameter below is nonzero, to specify that CS 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, CS Linux waits for the specified time before retrying the activation. If *react_timer_retry* is zero, this parameter is ignored.

ls_detail.def_data.react_timer_retry
Retry count for reactivating a failed LS. This parameter is used to specify whether CS Linux should attempt to reactivate the LS if it fails while in use (or if an attempt to start the LS fails).

Zero indicates that CS Linux should not attempt to reactivate the LS; a nonzero value specifies the number of retries to be made. A value of 65,535 indicates that CS Linux should retry indefinitely until the LS is activated.

CS Linux waits for the time specified by the *react_timer* parameter above between successive retries. If the retry count is reached without successfully reactivating the LS, or if a STOP_LS is issued while CS 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 AP_YES, the reactivation timer fields are ignored; if the link fails, CS Linux does not attempt to reactivate it until the user application that was using the session attempts to restart the session.

ls_detail.def_data.restart_on_normal_deact
Specifies whether CS Linux should attempt to reactivate the LS if it is deactivated normally by the remote system. Possible values are:

AP_YES
If the remote system deactivates the LS normally, CS 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).

AP_NO
If the remote system deactivates the LS normally, CS 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 AP_YES), this parameter is ignored; CS Linux always attempts to reactivate the LS if it is deactivated normally by the remote system (unless *react_timer_retry* is zero).

ls_detail.def_data.port_name
Name of the port associated with this link station. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 characters. If the link is to a VRN, this field specifies the name of the actual port used to connect to the VRN (as specified in the DEFINE_CN verb).

ls_detail.def_data.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.

The name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of 1-8 A-string characters, an EBCDIC dot (period) character, and a network name of 1-8 A-string characters.

ls_detail.def_data.adj_cp_type
Adjacent node type. This is one of the following:

AP_APPN_NODE
APPN-capable node; the node type will be learned during XID exchange.

AP_NETWORK_NODE
Network node, or Branch Network Node acting as a Network Node from the local node's perspective.

AP_END_NODE
End node, Branch Network Node acting as an End Node from the local node's perspective, or up-level LEN node (one that includes the Network Name CV in its XID3).

AP_BACK_LEVEL_LEN_NODE
Back-level LEN node (one that does not include the Network Name CV in its XID3).

AP_HOST_XID3
Host node; CS Linux responds to a polling XID from the node with a format 3 XID.

AP_HOST_XID0
Host node; CS Linux responds to a polling XID from the node with a format 0 XID.

AP_DSPU_XID
Downstream PU; CS Linux includes XID exchange in link activation.

AP_DSPU_NOXID
Downstream PU; CS Linux does not include XID exchange in link activation.

For SDLC:

ls_detail.def_data.dest_address
Address of the secondary link station.

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_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_PORT is a nonzero value), this parameter is set to either 0xFF to accept whatever address is configured at the secondary station, or 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_PORT is LS_PRI and *tot_link_act_lim* is greater than 1), this address must be different for each LS on the port.

For QLLC:

ls_detail.def_data.dest_address
Destination address of link station on the adjacent node. This parameter is used only for SVC outgoing calls (defined by the *vc_type* parameter in the link-specific data, and by the link activation limit parameters on DEFINE_PORT); it is ignored for incoming calls or for PVC.

For Token Ring, Ethernet:

ls_detail.def_data.dest_address.mac_address
MAC address of link station on adjacent node.

If this parameter is null, 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 Token Ring, the other Ethernet) connected by a bridge, you will probably need to reverse the bit order of the bytes in the MAC address. For more information, see “Bit ordering in MAC addresses” on page 124. If the two nodes are on the same LAN, or on LANs of the same type connected by a bridge, no change is required.

ls_detail.def_data.dest_address.lsap_address
Local SAP address of link station on adjacent node.

For multipath channel (MPC), CS Linux for IBM Z only:

def_data.dest_address.address
This parameter is reserved.

For Enterprise Extender (HPR/IP):

ls_detail.def_data.dest_address.ip_address_info.lsap
Local SAP address of link station on adjacent node. Specify a multiple of `0x04` in the range `0x04`-`0xEC`. The usual value is `0x04`, but VTAM may use `0x08` in some circumstances.

ls_detail.def_data.dest_address.ip_address_info.version
Specifies whether the following field represents an IPv4 or IPv6 address. Possible values:

IP_VERSION_4_HOSTNAME
The *address* field specifies an IPv4 address, or a hostname or alias that resolves to an IPv4 address.

IP_VERSION_6_HOSTNAME
The *address* field specifies an IPv6 address, or a hostname or alias that resolves to an IPv6 address.

ls_detail.def_data.dest_address.ip_address_info.address
IP address of link station on adjacent node. 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`).

For all link types:

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

AP_YES
The link can be activated automatically.

AP_NO
The link cannot be activated automatically.

ls_detail.def_data.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 AP_NETWORK_NODE or

AP_END_NODE); it is ignored otherwise. Zero indicates that the TG number is not preassigned and is negotiated when the link is activated.

ls_detail.def_data.limited_resource
Specifies whether this link station is to be deactivated when there are no sessions using the link. Possible values are:

AP_NO
The link is not a limited resource and will not be deactivated automatically.

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

AP_INACTIVITY
The link is a limited resource and will be deactivated automatically 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* field.

ls_detail.def_data.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 AP_NETWORK_NODE or AP_END_NODE); it is ignored otherwise. If the adjacent node is a host (*adj_cp_type* is either AP_HOST_XID3 or AP_HOST_XID0), CS Linux always requests the host to initiate SSCP sessions.

Possible values are:

AP_YES
Request the adjacent node to initiate SSCP sessions.

AP_NO
Do not request the adjacent node to initiate SSCP sessions.

ls_detail.def_data.pu_name
Name of the local PU that uses this link. This parameter is used only if *adj_cp_type* is set to AP_HOST_XID3 or AP_HOST_XID0, or if *solicit_sscp_sessions* is set to AP_YES; it is reserved otherwise.

The PU name is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces.

ls_detail.def_data.disable_remote_act
Specifies whether the LS can be activated by a remote node. Possible values are:

AP_YES
The LS can only be activated by the local node; if the remote node attempts to activate it, CS Linux will reject the attempt.

AP_NO
The LS can be activated by the remote node.

ls_detail.def_data.dspu_services
Specifies the services which the local node will provide 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 AP_NO; it is reserved otherwise. Possible values are:

AP_PU_CONCENTRATION
Local node will provide SNA gateway for the downstream PU.

AP_DLUR
Local node will provide DLUR services for the downstream PU.

AP_NONE
Local node will provide no services for this downstream PU.

ls_detail.def_data.dspu_name
Name of the downstream PU. This parameter is required if *solicit_sscp_sessions* is set to AP_NO and *dspu_services* is set to AP_PU_CONCENTRATION or AP_DLUR; it is reserved otherwise. The name is an 8-byte type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces.

ls_detail.def_data.dlus_name
Name of the DLUS node from which DLUR solicits SSCP services when the link to the downstream node is activated. This field is reserved if *dspu_services* is not set to AP_DLUR.

The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

A string of 17 binary zeros indicates the global default DLUS, defined using the DEFINE_DLUR_DEFAULTS verb. If this parameter is set to zeros and there is no global default DLUS, then DLUR will not initiate SSCP contact when the link is activated.

ls_detail.def_data.bkup_dlus_name
Name of the DLUS node from which DLUR solicits SSCP services when the link to the downstream node is activated. This field is reserved if *dspu_services* is not set to AP_DLUR.

The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

A string of 17 binary zeros indicates the global backup default DLUS, defined using the DEFINE_DLUR_DEFAULTS verb.

ls_detail.def_data.hpr_supported
Specifies whether HPR is supported on this link. Possible values are:

AP_YES
HPR is supported on this link.

AP_NO
HPR is not supported on this link.

ls_detail.def_data.hpr_link_lvl_error
For SDLC, this parameter is reserved.

For other port types, this parameter specifies whether link-level error recovery is supported for HPR traffic on the link.

This parameter is reserved if the *ls_detail.def_data.hpr_supported* parameter is set to AP_NO. Possible values are:

AP_YES
The HPR link-level error recovery tower is supported on this link.

AP_NO
The HPR link-level error recovery tower is not supported on this link.

ls_detail.def_data.link_deact_timer
Limited resource link deactivation timer (in seconds, minimum 5). If *limited_resource* is set to AP_INACTIVITY, the link will be deactivated if no data flows on it for the time specified by this parameter. A value of zero indicates no timeout (the link is not deactivated, as though *limited_resource* were set to AP_NO), and that values in the range 1-4 are interpreted as 5.

ls_detail.def_data.default_nn_server
End node: 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 LSs to find a suitable LS to activate. This allows 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:

AP_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.

AP_NO
This link should not be automatically activated in an attempt to contact a network node server.

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

ls_detail.def_data.ls_attributes
This array contains further information about the adjacent node, as described in the following parameters:

ls_detail.def_data.ls_attributes[0]
Host type (normally standard SNA). Possible values are:

AP_SNA
Standard SNA host.

AP_FNA
Fujitsu Network Architecture (VTAM-F) host.

AP_HNA
Hitachi Network Architecture host.

ls_detail.def_data.ls_attributes[1]
Network Name CV suppression for a link to a back-level LEN node.

If *adj_cp_type* is set to AP_BACK_LEVEL_LEN_NODE, this parameter specifies whether to suppress inclusion of the Network Name CV in the format 3 XID sent to the LEN node. Possible values are:

AP_NO
Include the Network Name CV in the XID.

AP_SUPPRESS_CP_NAME
Do not include the Network name CV.

If *adj_cp_type* is set to any other value, this parameter is reserved.

ls_detail.def_data.adj_node_id
Node ID of adjacent node. This is a 4-byte hexadecimal string; a value of 4 zeros indicates that node ID checking is disabled.

ls_detail.def_data.local_node_id
Node ID sent in XIDs on this LS. This is a 4-byte hexadecimal string, consisting of a block number (3 hexadecimal digits) and a node number (5 hexadecimal digits). A value of all zeros indicates that CS Linux uses the node ID specified in the DEFINE_NODE verb.

ls_detail.def_data.cp_cp_sess_support
Specifies whether CP-CP sessions are supported. Possible values are:

AP_YES
CP-CP sessions are supported.

AP_NO
CP-CP sessions are not supported.

ls_detail.def_data.use_default_tg_chars
Specifies whether the default TG characteristics supplied on the DEFINE_PORT verb are used. Possible values are:

AP_YES
Use the default TG characteristics; ignore the `tg_chars` structure on this verb.

AP_NO
Use the `tg_chars` structure on this verb.

ls_detail.def_data.tg_chars.effect_cap
Actual bits per second rate (line speed). The value is encoded as a 1-byte floating point number, represented by the formula 0.1 mmm * 2 ^{eeeee} where the bit representation of the byte is b'eeeeemmm'. Each unit of effective capacity is equal to 300 bits per second.

ls_detail.def_data.tg_chars.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.

ls_detail.def_data.tg_chars.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.

ls_detail.def_data.tg_chars.security
Security level of the network. Possible values are:

AP_SEC_NONSECURE
No security.

AP_SEC_PUBLIC_SWITCHED_NETWORK
Data is transmitted over a public switched network.

AP_SEC_UNDERGROUND_CABLE
Data is transmitted over secure underground cable.

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

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

AP_SEC_ENCRYPTED
Data is encrypted before transmission over the line.

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

AP_SEC_MAXIMUM
Maximum security.

ls_detail.def_data.tg_chars.prop_delay
Propagation delay: the time that a signal takes to travel the length of the link. Possible values are:

AP_PROP_DELAY_MINIMUM
Minimum propagation delay.

AP_PROP_DELAY_LAN
Delay is less than 480 microseconds (typical for a LAN).

AP_PROP_DELAY_TELEPHONE
Delay is in the range 480-49,512 microseconds (typical for a telephone network).

AP_PROP_DELAY_PKT_SWITCHED_NET
Delay is in the range 49,512-245,760 microseconds (typical for a packet-switched network).

AP_PROP_DELAY_SATELLITE
Delay is greater than 245,760 microseconds (typical for a satellite link).

AP_PROP_DELAY_MAXIMUM
Maximum propagation delay.

ls_detail.def_data.tg_chars.user_def_parm_1 **through** ***def_data.tg_chars.user_def_parm_3***
User-defined parameters, which include other TG characteristics not covered by the above parameters. Each of these parameters is set to a value in the range 0-255.

ls_detail.def_data.target_pacing_count
Numeric value between 1 and 32,767 inclusive indicating the desired pacing window size. (The current version of CS Linux does not make use of this value.)

ls_detail.def_data.max_send_btu_size
Maximum BTU size that can be sent.

ls_detail.def_data.ls_role
Link station role. This is normally set to AP_USE_PORT_DEFAULTS, specifying that the LS role is taken from the definition of the port that owns this LS.

If the LS has been defined with a specific LS role overriding the port definition, this is one of the following values:

AP_LS_PRI
Primary

AP_LS_SEC
Secondary

AP_LS_NEG
Negotiable

ls_detail.def_data.max_ifrm_rcvd
Maximum of I-frames that can be received by this link station before an acknowledgment is sent. This value is in the range 0-127. When this field is zero, the value of *max_ifrm_rcvd* from DEFINE_PORT is used as default.

ls_detail.def_data.dlus_retry_timeout
Interval in seconds between second and subsequent attempts to contact the DLUS specified in the *ls_detail.def_data.dlus_name* and *ls_detail.def_data.bkup_dlus_name* parameters. The interval between the initial attempt and the first retry is always one second. If zero is returned, the default value configured with DEFINE_DLUR_DEFAULTS is used. This parameter is ignored if *ls_detail.def_data.dspu_services* is not set to AP_DLUR.

ls_detail.def_data.dlus_retry_limit
Maximum number of retries after an initial failure to contact the DLUS specified in the *ls_detail.def_data.dlus_name* and *ls_detail.def_data.bkup_dlus_name* parameters. If zero is returned, the default value configured through DEFINE_DLUR_DEFAULTS is used. If 0x0FFFF is returned, CS Linux retries indefinitely. This parameter is ignored if *ls_detail.def_data.dspu_services* is not set to AP_DLUR.

def_data.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:

AP_YES
Data compression should be used for LU 0-3 sessions on this link if the host requests it.

AP_NO
Data compression should not be used for LU 0-3 sessions on this link.

ls_detail.def_data.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 *def_data.adj_cp_type* is set to AP_NETWORK_NODE, AP_END_NODE, AP_APPN_NODE, or AP_BACK_LEVEL_LEN_NODE, this parameter defines whether the link is an uplink or a downlink. Possible values are:

AP_UPLINK
The link is an uplink.

AP_DOWNLINK
The link is a downlink.

ls_detail.def_data.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 *def_data.adj_cp_type* is set to AP_NETWORK_NODE, or it is set to AP_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:

AP_BRNN_ALLOWED
The adjacent node is allowed (but not required) to be a Branch Network Node.

AP_BRNN_REQUIRED
The adjacent node must be a Branch Network Node.

AP_BRNN_PROHIBITED
The adjacent node must not be a Branch Network Node.

ls_detail.def_data.pu_can_send_dddlu_offline
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), CS 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 AP_YES and *dspu_services* is not set to AP_NONE).

Possible values are:

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

AP_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 AP_NO.

ls_detail.def_data.link_spec_data_len
Length of link-specific data that is passed unchanged to link station component during initialization. The data structure for this data follows the `ls_def_data` structure, but is padded to start on a 4-byte boundary. For more details of the link-specific data, see “DEFINE_LS” on page 104.

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_LINK_NAME
The *list_options* parameter was set to AP_LIST_INCLUSIVE to list all entries starting from the supplied name, but the *ls_name* parameter was not valid.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

照会 LS_ルーティング

QUERY_LS_ROUTING verb は、リンク・ステーションを使用するパートナー LU の位置についてのローカル LU に関する情報を戻します。 複数のローカル LU に関する情報が要求された場合、その情報はローカ

ル LU 名によって順序付けされ、その後、各ローカル LU 名に関連付けられたパートナー LU 名によって順序付けられます。 ワイルドカード・パートナー LU 名を、ワイルドカードを含まない項目と混在させることができます。

VCB structure

```
typedef struct query_ls_routing
{
  AP_UINT16         opcode;               /* verb operation code           */
  unsigned char     reserv2;              /* reserved                      */
  unsigned char     format;               /* reserved                      */
  AP_UINT16         primary_rc;           /* primary return code           */
  AP_UINT32         secondary_rc;         /* secondary return code         */
  unsigned char    *buf_ptr;              /* buffer pointer                */
  AP_UINT32         buf_size;             /* buffer size                   */
  AP_UINT32         total_buf_size;       /* total buffer size             */
  AP_UINT16         num_entries;          /* number of entries             */
  AP_UINT16         total_num_entries;    /* total number of entries       */
  unsigned char     list_options;         /* list options                  */
  unsigned char     reserv3;              /* reserved                      */
  unsigned char     lu_name[8];           /* LU Name                       */
  unsigned char     lu_alias[8];          /* reserved                      */
  unsigned char     fq_partner_lu[17];    /* partner lu name               */
  unsigned char     wildcard_fqplu;       /* wildcard partner LU flag      */
  unsigned char     reserv4[2];           /* reserved                      */
} QUERY_LS_ROUTING;
```

```
typedef struct ls_routing_data
{
  AP_UINT16        overlay_size;
  unsigned char    lu_name[8];             /* local LU name                */
  unsigned char    lu_alias[8];            /* reserved                     */
  unsigned char    fq_partner_lu[17];      /* partner lu                   */
  unsigned char    wildcard_fqplu;         /* wildcard partner LU flag     */
  unsigned char    ls_name[8];             /* link to use                  */
  unsigned char    reserv3[2];             /* reserved                     */
} LS_ROUTING_DATA;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
照会 LS_LS_ルーティング

num_entries
データが戻される LS 経路指定項目の最大数。 ある範囲ではなく、特定の LS のデータを要求するには、値 1 を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータの戻りを開始する LS 経路指定項目のリスト内の位置。

次の値のいずれかを指定してください。

リストの最初のリスト (_R)
リストの最初の項目から開始します。

リストを含む (包括的)
lu_name パラメーターと *fq_partner_lu* パラメーターの組み合わせによって指定されたエントリーから開始します。

次への AP_LIST_FROM_NEXT
lu_name、*fq_partner_lu*、および ワイルドカード・ディスク パラメーターの組み合わせによって指定されたエントリーの直後のエントリーから開始します。

リストの順序とアプリケーションでの特定のエントリーの取得方法について詳しくは、34 ページの『List options for QUERY_* Verbs』を参照してください。

lu_name
CS Linux に定義されているローカル LU の名前。 これは 8 バイトのタイプ A の EBCDIC ストリングで、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。 *list_options* を **リストの最初のリスト** （ _R ）に設定すると、このパラメーターは無視されます。

lu_alias
このパラメーターは予約されており、 2 進ゼロに設定します。

fq_partner_lu
CS Linux に対して定義されている、パートナー LU の完全修飾名。 この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 このパラメーターは、指定されたローカル LU のパートナー LU 名のリスト内で戻される項目を修飾するために使用されます。 *list_options* を **リストの最初のリスト** （ _R ）に設定すると、このパラメーターは無視されます。

このパラメーターが 2 進ゼロに設定され、 *list_options* が **次への** AP_LIST_FROM_NEXT に設定されている場合、返されるリストは、 *lu_name* パラメーターによって識別される LU の最初のパートナー LU 名から開始されます。

ワイルドカード・ディスク
ワイルドカード・パートナー LU フラグ。 *fq_partner_lu* パラメーターに完全ワイルドカードまたは部分ワイルドカードが含まれているかどうかを示します。 このフラグは、戻す最初のレコードを識別するためにのみ使用されます。 これを使用して、ワイルドカード指定に一致するエントリーのみを戻すように指定することはできません。 可能な値は次のとおりです

類人猿
fq_partner_lu パラメーターにワイルドカード・エントリーが含まれています。

アプ・ノー
fq_partner_lu パラメーターにワイルドカード・エントリーが含まれていません。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

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:

AP_YES
The *fq_partner_lu* parameter contains a full or partial wildcard.

AP_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.

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

ファイル名の変更
list_options パラメーターが **リストを含む （ 包括的 ）**に設定されていましたが、 *lu_name* paraemter に指定された値が、どの exisiting LS ルーティング・データ・レコードとも一致しませんでした。

ファイル名の入力が無効です
list_options パラメーターが **リストを含む （ 包括的 ）**に設定されましたが、 *fq_partner_lu* パラメーターによって指定された値が、指定されたパートナー LU 名の既存の LS 経路指定データ・レコードと一致しませんでした。

ファイル・ワイルドカード名を使用できます
ワイルドカード・ディスク パラメーターが **類人猿**に設定されましたが、 *fq_partner_lu* パラメーターが有効なワイルドカード名ではありませんでした。

ファイルの追加リスト・オプション
list_options パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

照会 LU_0_TO_3

QUERY_LU_0_TO_3 は、タイプ 0、 1、 2、または 3 のローカル LU に関する情報を戻します。 この情報は、 "決定データ" (実行中に動的に収集されたデータ、ノードがアクティブの 場合にのみ戻されるデータ) および "定義済みデータ" (DEFINE_LU_0_TO_3 でアプリケーションによって提供されるデータ) として構造化されます。

この verb は、使用されるオプションに応じて、特定の LU または複数の LU に関する要約情報または詳細情報を取得するために使用できます。

VCB 構造体

```
typedef struct query_lu_0_to_3
{
   AP_UINT16       opcode;                 /* verb operation code          */
   unsigned char   reserv2;                /* reserved                     */
   unsigned char   format;                 /* reserved                     */
   AP_UINT16       primary_rc;             /* primary return code          */
   AP_UINT32       secondary_rc;           /* secondary return code        */
   unsigned char   *buf_ptr;               /* pointer to buffer            */
   AP_UINT32       buf_size;               /* buffer size                  */
   AP_UINT32       total_buf_size;         /* total buffer size required   */
   AP_UINT16       num_entries;            /* number of entries            */
   AP_UINT16       total_num_entries;      /* total number of entries      */
   unsigned char   list_options;           /* listing options              */
   unsigned char   reserv3;                /* reserved                     */
   unsigned char   pu_name[8];             /* PU name filter               */
   unsigned char   lu_name[8];             /* LU name                      */
   unsigned char   host_attachment;        /* host attachment filter       */
} QUERY_LU_0_TO_3;
```

```
typedef struct lu_0_to_3_summary
{
   AP_UINT16       overlay_size;           /* size of returned entry       */
   unsigned char   pu_name[8];             /* PU name                      */
   unsigned char   lu_name[8];             /* LU name                      */
   unsigned char   description[32];        /* resource description         */
   unsigned char   reserv1[16];            /* reserved                     */
   unsigned char   nau_address;            /* NAU address                  */
   unsigned char   lu_sscp_sess_active;    /* Is LU-SSCP session active    */
   unsigned char   appl_conn_active;       /* Is connection to appl active */
   unsigned char   plu_sess_active;        /* Is PLU-SLU session active    */
```

```
   unsigned char   host_attachment;          /* LU's host attachment        */
} LU_0_TO_3_SUMMARY;
```

```
typedef struct lu_0_to_3_detail
{
   AP_UINT16       overlay_size;             /* size of returned entry      */
   unsigned char   lu_name[8];               /* LU name                     */
   unsigned char   reserv1[2];               /* reserved                    */
   LU_0_TO_3_DET_DATA  det_data;             /* Determined data             */
   LU_0_TO_3_DEF_DATA  def_data;             /* Defined data                */
} LU_0_TO_3_DETAIL;
```

```
typedef struct lu_0_to_3_det_data
{
   unsigned char   lu_sscp_sess_active;      /* Is LU-SSCP session active   */
   unsigned char   appl_conn_active;         /* Application is using LU     */
   unsigned char   plu_sess_active;          /* Is PLU-SLU session active   */
   unsigned char   host_attachment;          /* Host attachment             */
   SESSION_STATS   lu_sscp_stats;            /* reserved                    */
   SESSION_STATS   plu_stats;                /* reserved                    */
   unsigned char   plu_name[8];              /* PLU name                    */
   unsigned char   session_id[8];            /* Internal ID of PLU-SLU sess */
   unsigned char   app_spec_det_data[360];   /* Application specified data  */
   unsigned char   app_type;                 /* Type of application using LU */
   unsigned char   sscp_id[6];               /* sscp id                     */
   unsigned char   bind_lu_type;             /* LU type from BIND message   */
   unsigned char   compression;              /* data compression level      */
   unsigned char   cryptography;             /* reserved                    */
   unsigned char   reserva[10];              /* reserved                    */
} LU_0_TO_3_DET_DATA;
```

```
typedef struct session_stats
{
   AP_UINT16       rcv_ru_size;          /* session receive RU size         */
   AP_UINT16       send_ru_size;         /* session send RU size            */
   AP_UINT16       max_send_btu_size;    /* maximum send BTU size           */
   AP_UINT16       max_rcv_btu_size;     /* maximum rcv BTU size            */
   AP_UINT16       max_send_pac_win;     /* maximum send pacing window size */
   AP_UINT16       cur_send_pac_win;     /* current send pacing window size */
   AP_UINT16       max_rcv_pac_win;      /* maximum receive pacing window   */
                                         /*  size                           */
   AP_UINT16       cur_rcv_pac_win;      /* current receive pacing window   */
                                         /*  size                           */
   AP_UINT32       send_data_frames;     /* number of data frames sent      */
   AP_UINT32       send_fmd_data_frames; /* num fmd data frames sent        */
   AP_UINT32       send_data_bytes;      /* number of data bytes sent       */
   AP_UINT32       rcv_data_frames;      /* number of data frames received  */
   AP_UINT32       rcv_fmd_data_frames;  /* num fmd data frames received    */
   AP_UINT32       rcv_data_bytes;       /* number of data bytes received   */
   unsigned char   sidh;                 /* session ID high byte (from LFSID)*/
   unsigned char   sidl;                 /* session ID low byte (from LFSID) */
   unsigned char   odai;                 /* ODAI bit set                    */
   unsigned char   ls_name[8];           /* Link station name               */
   unsigned char   pacing_type;          /* type of pacing in use           */
} SESSION_STATS;
```

```
typedef struct lu_0_to_3_def_data
{
   unsigned char   description[32];          /* resource description        */
   unsigned char   reserv1[16];              /* reserved                    */
   unsigned char   nau_address;              /* LU NAU address              */
   unsigned char   pool_name[8];             /* LU Pool name                */
   unsigned char   pu_name[8];               /* PU name                     */
   unsigned char   priority;                 /* LU priority                 */
   unsigned char   lu_model;                 /* LU model (type)             */
   unsigned char   sscp_id[6];               /* SSCP ID                     */
   AP_UINT16       timeout;                  /* Timeout                     */
   unsigned char   app_spec_def_data[16];    /* application-specified data  */
   unsigned char   model_name[7];            /* reserved                    */
   unsigned char   term_method;              /* session termination type    */
    unsigned char       disconnect_on_unbind;    /* disconnect on UNBIND flag */
   unsigned char   reserv3[15];              /* reserved                    */
} LU_0_TO_3_DEF_DATA;
```

lu_0_to_3_det_data 構造体内の アプリケーション・タイプ パラメーターが AP_LUA_APPLICATION に設定されている場合、 *app_spec_det_data* データ フィールドには以下の構造が含まれます。

```
typedef struct lua_session_user_info
{
   unsigned char   user_ip_address[40];      /* IP address of LUA application    */
   unsigned char   user_host_address[256];   /* Host name of LUA application     */
   unsigned char   reserved[24];             /* reserved                         */
} SESSION_USER_INFO;
```

lu_0_to_3_det_data 構造体内の アプリケーション・タイプ パラメーターが AP_FMI_アプリケーションに設定されている場合、 *app_spec_det_data* データ フィールドには以下の構造が含まれます。

```
typedef struct session_user_info
{
   unsigned char   user_name[32];            /* 3270 user name               */
   unsigned char   system_name[128];         /* computer name                */
   AP_UINT32       user_pid;                 /* process ID                   */
   AP_UINT32       user_type;                /* type of application using LU */
   AP_UINT32       user_uid;                 /* user ID                      */
   AP_UINT32       user_gid;                 /* group ID                     */
   unsigned char   user_gname[32];           /* group name                   */
   unsigned char   reserv4[32];              /* reserved                     */
} SESSION_USER_INFO;
```

lu_0_to_3_det_data 構造体の アプリケーション・タイプ パラメーターが **付加プールの集中**に設定されている場合、 *app_spec_det_data* データ フィールドには、 アプリケーション・タイプ パラメーターが **付加プールの集中** に設定されており、 ユーザー名 から ユーザー名 のパラメーターが *pu_conc_downstream_lu* パラメーターによって置き換えられる点を除いて、上記の 3270 構造と同じ構造が含まれます。

lu_0_to_3_det_data 構造内の アプリケーション・タイプ パラメーターが AP_LUA_APPLICATION に設定されている場合、 *app_spec_det_data* データ フィールドには、 アプリケーション・タイプ パラメーターが AP_LUA_APPLICATION に設定されており、 ユーザー名 から ユーザー名 までのパラメーターが戻されないことを除いて、上記の 3270 構造と同じ構造が含まれます。

lu_0_to_3_det_data 構造体内の アプリケーション・タイプ パラメーターが **アプリケーション・ サーバーの追加**に設定されている場合、 *app_spec_det_data* データ フィールドには以下の構造が含まれます。

```
typedef struct tn_server_session_user_info
{
   unsigned char   user_ip_address[40];      /* user's IP address            */
   AP_UINT16       port_number;              /* TCP/IP port number           */
   AP_UINT16       cb_number;                /* reserved                     */
   AP_UINT16       cfg_default;              /* using the default record?    */
   unsigned char   cfg_address[68];          /* address from config record   */
   AP_UINT16       cfg_format;               /* format of address            */
   unsigned char   tn3270_level;             /* TN3270 level used:           */
                                             /*   AP_LEVEL_TN3270            */
                                             /*   AP_LEVEL_TN3270E           */
   unsigned char   lu_select;                /* method of LU selection:      */
                                             /*   AP_GENERIC_LU              */
                                             /*   AP_SPECIFIC_LU             */
                                             /*   AP_ASSOCIATED_LU           */
   unsigned char   request_lu_name[8];       /* requested LU name or         */
                                             /* associated display LU name   */
                                             /*  (in EBCDIC)                 */
   unsigned char   cipher_spec;              /* SSL cipher specification     */
   unsigned char   reserv3[21];              /* reserved                     */
} TN_SERVER_SESSION_USER_INFO;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_LU_0_TO_3

buf_ptr
A pointer to a data buffer that CS Linux will use to return the requested information.

buf_size
Size of the supplied data buffer.

num_entries
Maximum number of LUs for which data should be returned. To request data for a specific LU rather than a range, specify the value 1. To return as many entries as possible, specify 0; in this case, CS Linux will return the maximum number of entries that can be accommodated in the supplied data buffer.

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

AP_SUMMARY
Summary information only.

AP_DETAIL
Detailed information.

Combine this value using a logical OR operation with one of the following values:

AP_FIRST_IN_LIST
Start at the first entry in the list.

AP_LIST_INCLUSIVE
Start at the entry specified by the *lu_name* parameter.

AP_LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *lu_name* parameter.

For more information about how the list is ordered and how the application can obtain specific entries from it, see “List options for QUERY_* Verbs” on page 34.

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, set this field to binary zeros.

lu_name
Name of the local LU. This is an 8-byte type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces if the name is shorter than 8 characters. This parameter is ignored if *list_options* is set to AP_FIRST_IN_LIST.

host_attachment
Host attachment filter. If the verb 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:

AP_DIRECT_ATTACHED
Return information only on LUs directly attached to the host system.

AP_DLUR_ATTACHED
Return information only on LUs supported by DLUR on the local node.

AP_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.

AP_PU_CONCENTRATION
Return information only on LUs supported by SNA gateway from a downstream node.

AP_NONE
Return information about all LUs regardless of host attachment.

If the node is not running, this parameter is ignored; CS Linux returns information about all LUs regardless of host attachment.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

lu_0_to_3_summary.overlay_size
The size of the returned `lu_0_to_3_summary` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `lu_0_to_3_summary` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

lu_0_to_3_summary.pu_name
Name of the local PU used by the LU. This is an 8-byte type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces if the name is shorter than 8 characters.

lu_0_to_3_summary.lu_name
Name of the local LU. This is an 8-byte type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces if the name is shorter than 8 characters.

lu_0_to_3_summary.description
A null-terminated text string describing the LU, as specified in the definition of the LU.

lu_0_to_3_summary.nau_address
Network accessible unit address of the LU. This is in the range 1-255.

lu_0_to_3_summary.lu_sscp_sess_active
Specifies whether the LU-SSCP session is active. Possible values are:

AP_YES
The session is active.

AP_NO
The session is inactive.

lu_0_to_3_summary.appl_conn_active
Specifies whether an application is using the LU. Possible values are:

AP_YES
An application is using the LU.

AP_NO
No application is using the LU.

lu_0_to_3_summary.plu_sess_active
Specifies whether the PLU-SLU session is active. Possible values are:

AP_YES
The session is active.

AP_NO
The session is inactive.

lu_0_to_3_summary.host_attachment
LU host attachment type.

When the verb is issued to a running node, this parameter takes one of the following values:

AP_DIRECT_ATTACHED
LU is directly attached to the host system.

AP_DLUR_ATTACHED
LU is supported by DLUR on the local node.

AP_DLUR
LU is supported by passthrough DLUR from a downstream node.

AP_PU_CONCENTRATION
LU is supported by SNA gateway from a downstream node.

lu_0_to_3_detail.overlay_size
The size of the returned `lu_0_to_3_detail` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `lu_0_to_3_detail` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

lu_0_to_3_detail.lu_name
Name of the local LU. This is an 8-byte type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces if the name is shorter than 8 characters.

lu_0_to_3_detail.det_data.lu_sscp_sess_active
Specifies whether the LU-SSCP session is active. Possible values are:

AP_YES
The session is active.

AP_NO
The session is inactive.

lu_0_to_3_detail.det_data.appl_conn_active
Specifies whether an application is using the LU. Possible values are:

AP_YES
An application is using the LU.

AP_NO
No application is using the LU.

lu_0_to_3_detail.det_data.plu_sess_active
Specifies whether the PLU-SLU session is active. Possible values are:

AP_YES
The session is active.

AP_NO
The session is inactive.

lu_0_to_3_detail.det_data.host_attachment
LU host attachment type.

When the verb is issued to a running node, this parameter takes one of the following values:

AP_DIRECT_ATTACHED
LU is directly attached to the host system.

AP_DLUR_ATTACHED
LU is supported by DLUR on the local node.

AP_DLUR
LU is supported by passthrough DLUR from a downstream node.

AP_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 `session_stats` structure contains the following parameters:

rcv_ru_size
Maximum receive RU size. (In the LU-SSCP session statistics, this parameter is reserved.)

send_ru_size
Maximum send RU size. (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.

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 associated with statistics. This is an 8-byte ASCII character string, right-padded with spaces if the name is shorter than 8 characters.

pacing_type
Receive pacing type in use on the PLU-SLU session. Possible values are:

- AP_NONE
- AP_PACING_FIXED

lu_0_to_3_detail.det_data.plu_name
Name of the primary LU. This is an 8-byte type-A EBCDIC string, right-padded with spaces if the name is shorter than 8 characters. This name is reserved if the PLU-SLU session is inactive.

lu_0_to_3_detail.det_data.session_id
Eight byte internal identifier of the PLU-SLU session.

lu_0_to_3_detail.det_data.app_spec_det_data
The format of the data in this field depends on the value of the *app_type* field below, as follows:

- If *app_type* is set to AP_NONE, this field is reserved.
- If *app_type* is set to AP_PU_CONCENTRATION, the first 8 bytes of this field contain the LU name of the downstream LU currently using this local LU. This is an EBCDIC string, right-padded with spaces if the name is shorter than 8 characters. The remaining bytes are reserved.
- If *app_type* is set to AP_LUA_APPLICATION, this field is replaced by the lua_session_user_info structure, as described below.
- If *app_type* is set to AP_FMI_APPLICATION, this field is replaced by the session_user_info structure, as described below.

If *app_type* is set to AP_LUA_APPLICATION, the *app_spec_det_data*field is replaced by the lua_session_user_info structure, containing information about the LUA application using this LU. The structure consists of the following fields:

user_ip_address
The IP address of the computer (client or server) where the LUA application 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).

user_host_address
The name of the computer (client or server) where the LUA application is running. This is a null-terminated ASCII string, representing an IP hostname (such as newbox.this.co.uk).

If *app_type* is set to AP_FMI_APPLICATION, the *app_spec_det_data*field is replaced by the session_user_info structure, containing information about the user of this LU. The structure consists of the following fields:

user_name
The Linux user name with which the 3270 emulation program using this LU is running. This is an ASCII string of 1-32 characters.

system_name
The computer name on which the program is running.

user_pid
The process ID of the program using the LU.

user_type
The type of session (3270 display session, 3270 printer session) using the LU. Possible values are:

AP_3270_DISPLAY_MODEL_2

AP_3270_DISPLAY_MODEL_3

AP_3270_DISPLAY_MODEL_4

AP_3270_DISPLAY_MODEL_5

AP_PRINTER

AP_SCS_PRINTER

AP_UNKNOWN

user_uid
The Linux user ID with which the program is running.

user_gid
The Linux group ID with which the program is running.

user_gname
The Linux group name with which the program is running. This is an ASCII string of 1-32 characters.

If *app_type* is set to AP_TN_SERVER, this field is replaced by the tn_server_session_user_info structure, containing information about the TN3270 program that is using this LU. The structure consists of the following fields:

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 177. Possible values are:

AP_YES
The program is using the default record. The *cfg_address* and *cfg_format* parameters below are reserved.

AP_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. This can be any of the following; the format is indicated by the *cfg_format* parameter.

- 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).

cfg_format
Specifies the format of the *cfg_address* parameter. Possible values are:

AP_ADDRESS_IP
IP address

AP_ADDRESS_FQN
Alias or fully qualified name

tn3270_level
Level of TN3270 support. Possible values are:

AP_LEVEL_TN3270
TN3270E protocols are disabled.

AP_LEVEL_TN3270E
TN3270E protocols are enabled.

lu_select
Method of LU selection. Possible values are:

AP_GENERIC_LU
The TN3270 program selected a generic display or printer LU.

AP_SPECIFIC_LU
The TN3270 program selected this LU specifically.

AP_ASSOCIATED_LU
This is a printer LU that has been associated with a display LU by a DEFINE_TN3270_ASSOCIATION verb, or a display LU that has been associated with a printer LU by a DEFINE_TN3270_ASSOCIATION verb. The LU is in use by the TN3270 through its association.

request_lu_name
Requested LU name or associated display LU name. This is an 8-byte type-A EBCDIC string, padded on the right with spaces if the name is shorter than 8 characters.

cipher_spec
Indicates the type of SSL security and the encryption level in use for this session. Possible values are:

AP_SSL_NO_SSL
SSL is not being used.

AP_TLS_RSA_WITH_NULL_NULL AP_TLS_RSA_WITH_NULL_SHA AP_TLS_RSA_WITH_NULL_MD5
Certificates are exchanged, but no encryption is used.

AP_TLS_RSA_WITH_DES_CBC_SHA
DES 56-bit encryption (deprecated).

AP_TLS_AES_128_CCM_8_SHA256 AP_TLS_AES_128_CCM_SHA256
AP_TLS_AES_128_GCM_SHA256 AP_TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
AP_TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
AP_TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
AP_TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
AP_TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
AP_TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 AP_TLS_RSA_WITH_AES_128_CBC_SHA
AP_TLS_RSA_WITH_AES_128_CBC_SHA256 AP_TLS_RSA_WITH_AES_128_GCM_SHA256
128-bit encryption.

AP_TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
AP_TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA AP_TLS_RSA_WITH_3DES_EDE_CBC_SHA
Triple-DES 168-bit encryption.

AP_TLS_AES_256_GCM_SHA384 AP_TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
AP_TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
AP_TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
AP_TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
AP_TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
AP_TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 AP_TLS_RSA_WITH_AES_256_CBC_SHA
AP_TLS_RSA_WITH_AES_256_CBC_SHA256 AP_TLS_RSA_WITH_AES_256_GCM_SHA384
AP_TLS_CHACHA20_POLY1305_SHA256
256-bit encryption (recommended).

lu_0_to_3_detail.det_data.app_type
The type of application, if any, that is using the LU. Possible values are:

AP_NONE
The LU is not in use.

AP_LUA_APPLICATION
The LU is being used by an LUA application.

AP_PU_CONCENTRATION
The LU is being used by a downstream LU using SNA gateway.

AP_FMI_APPLICATION
The LU is being used by a 3270 emulation program.

AP_TN_SERVER
The LU is being used by a TN3270 program accessing TN server.

lu_0_to_3_detail.det_data.sscp_id
A 6-byte field containing the SSCP ID received in the ACTPU for the PU used by this LU. If *lu_sscp_sess_active* is AP_NO, this parameter will be all zeros.

lu_0_to_3_detail.det_data.bind_lu_type
Specifies the LU type of the LU which issued the original BIND (if there is an active LU-LU session). Possible values are:

AP_LU_TYPE_0
LU type 0.

AP_LU_TYPE_1
LU type 1.

AP_LU_TYPE_2
LU type 2.

AP_LU_TYPE_3
LU type 3.

AP_LU_TYPE_6
Downstream dependent LU 6.2.

AP_LU_TYPE_UNKNOWN
There is no active LU-LU session.

lu_0_to_3_detail.det_data.compression
Compression level in use on the PLU-SLU session, if any. Possible values are:

AP_NO
Data flowing on the PLU-SLU session is not compressed by CS Linux, or there is no active PLU-SLU session.

AP_YES
CS 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.

AP_PASSTHRU
Compression on this session is performed by the session endpoints (the host LU and the local application or downstream LU), and not by CS Linux.

lu_0_to_3_detail.def_data.description
A null-terminated text string describing the LU, as specified in the definition of the LU.

lu_0_to_3_detail.def_data.nau_address
Network accessible unit address of the LU, in the range 1-255.

lu_0_to_3_detail.def_data.pool_name
Name of the LU pool to which this LU belongs. This is an 8-byte EBCDIC string, padded on the right with spaces if the name is shorter than 8 characters. If the LU does not belong to a pool, this field is set to 8 binary zeros.

lu_0_to_3_detail.def_data.pu_name
Name of the PU (as specified on the DEFINE_LS verb) which this LU will use. This is an 8-byte type-A EBCDIC string, padded on the right with spaces if the name is shorter than 8 characters.

lu_0_to_3_detail.def_data.priority
LU priority when sending to the host. This is set to one of the following:

- AP_NETWORK
- AP_HIGH
- AP_MEDIUM
- AP_LOW

lu_0_to_3_detail.def_data.lu_model
Type of the LU. This is set to one of the following:

AP_3270_DISPLAY_MODEL_2

AP_3270_DISPLAY_MODEL_3

AP_3270_DISPLAY_MODEL_4

AP_3270_DISPLAY_MODEL_5

AP_PRINTER

AP_SCS_PRINTER

AP_UNKNOWN

lu_0_to_3_detail.def_data.sscp_id
Specifies the ID of the SSCP permitted to activate this LU. This is a 6-byte binary field. If this parameter is set to binary zeros, the LU may be activated by any SSCP.

lu_0_to_3_detail.def_data.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.

lu_0_to_3_detail.def_data.term_method
This parameter specifies how CS Linux should attempt to end a PLU-SLU session to a host from this LU. Possible values are:

AP_USE_NODE_DEFAULT
Use the node's default termination method, specified by the *send_term_self* parameter on DEFINE_NODE.

AP_SEND_UNBIND
End the session by sending an UNBIND.

AP_SEND_TERM_SELF
End the session by sending a TERM_SELF.

lu_0_to_3_detail.def_data.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:

AP_YES
End the session if the host sends an UNBIND that is not type 2 (BIND forthcoming).

AP_NO
Do not end the session if the host sends an UNBIND.

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_LU_NAME
The *list_options* parameter was set to AP_LIST_INCLUSIVE to list all entries starting from the supplied name, but the *lu_name* parameter was not valid.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

照会 LU_LU_PASSWORD

QUERY_LU_LU_PASSWORD は、ローカル LU とパートナー LU の間のセッション・レベル・セキュリティー検査に使用されるパスワードに関する情報を戻します。 これは、使用されるオプションに応じて、特定のパートナー LU のパスワードまたは複数のパートナー LU のパスワードに関する情報を取得するために使用できます。

VCB 構造体

```
typedef struct query_lu_lu_password
{
   AP_UINT16        opcode;                 /* Verb operation code          */
   unsigned char    reserv2;                /* reserved                     */
   unsigned char    format;                 /* reserved                     */
   AP_UINT16        primary_rc;             /* Primary return code          */
   AP_UINT32        secondary_rc;           /* Secondary return code        */
   unsigned char    *buf_ptr;               /* pointer to buffer            */
   AP_UINT32        buf_size;               /* buffer size                  */
   AP_UINT32        total_buf_size;         /* total buffer size required   */
   AP_UINT16        num_entries;            /* number of entries    *       */
   AP_UINT16        total_num_entries;      /* total number of entries      */
   unsigned char    list_options;           /* listing options              */
   unsigned char    reserv3;                /* reserved                     */
   unsigned char    lu_name[8];             /* LU name                      */
   unsigned char    lu_alias[8];            /* LU alias                     */
   unsigned char    plu_alias[8];           /* partner LU alias             */
   unsigned char    fqplu_name[17];         /* fully-qual. partner LU name  */
} QUERY_LU_LU_PASSWORD;
```

```
typedef struct password_info
{
   AP_UINT16        overlay_size;           /* size of returned entry       */
   unsigned char    plu_alias[8];           /* partner LU alias             */
   unsigned char    fqplu_name[17];         /* fully-qual. partner LU name  */
   unsigned char    description[32];        /* resource description         */
   unsigned char    reserv1[16];            /* reserved                     */
   unsigned char    password[8];            /* password                     */
   unsigned char    protocol_defined;       /* protocol defined             */
   unsigned char    protocol_in_use;        /* protocol in use              */
} PASSWORD_INFO;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
パスワードの追加を照会

buf_ptr
CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

num_entries
パスワード情報が戻されるパートナー LU の最大数。 特定の範囲ではなく特定の項目を要求するには、 1 という値を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、 CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータの戻りを開始する必要があるリスト内の位置。 次の値のいずれかを指定してください。

リストの最初のリスト （ _R ）
リストの最初の項目から開始します。

リストを含む （ 包括的 ）
plu_alias パラメーターまたは *fqplu_name* パラメーターで指定したエントリーから開始します。

次への AP_LIST_FROM_NEXT
plu_alias パラメーターまたは *fqplu_name* パラメーターによって指定されたエントリーの直後のエントリーから開始します。

リストの順序とアプリケーションでの特定のエントリーの取得方法について詳しくは、 34 ページの『List options for QUERY_* Verbs』を参照してください。

lu_name
LU 名。 この名前は、 8 バイトのタイプ A の EBCDIC 文字ストリングです。 LU がその LU 名ではなく LU 別名によって識別されることを示すには、このパラメーターを 8 桁の 2 進ゼロに設定し、 *lu_alias* パラメーターに LU 別名を指定します。

lu_alias
ローカルに定義された LU の別名。 これは、 8 バイトの ASCII 文字ストリングです。 このパラメーターが使用されるのは、 *lu_name* がすべてゼロに設定されている場合のみです。それ以外の場合は CP (デフォルトの LU) に関連した LU を示すためには、 *lu_name* と *lu_alias* の両方をすべてゼロに設定します。

plu_alias
パートナー LU の別名。 これは、 8 バイトの ASCII 文字ストリングです。 *list_options* が **リストの最初のリスト** （ _R ）に設定されている場合、このパラメーターは無視されます。それ以外の場合は、パートナー LU の LU 別名または完全修飾 LU 名のいずれかを指定する必要があります。 パートナー LU が LU 別名の代わりに完全修飾 LU 名によって識別されることを示すには、このパラメーターを 8 桁の 2 進ゼロに設定し、 *fqplu_name* パラメーターに LU 別名を指定します。

fqplu_name
パートナー LU の完全修飾ネットワーク名。 *list_options* が **リストの最初のリスト** （ _R ）に設定されている場合、このパラメーターは無視されます。それ以外の場合は、パートナー LU の LU 別名または完全修飾 LU 名のいずれかを指定する必要があります。 このパラメーターが使用されるのは、 *plu_alias* がすべてゼロに設定されている場合のみです。それ以外の場合は

この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

password_info.overlay_size
The size of the returned `password_info` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `password_info` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

password_info.plu_alias
Partner LU alias. This is an 8-byte ASCII character string.

password_info.fqplu_name
Fully qualified network name for the partner LU. The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

password_info.description
A null-terminated text string describing the LU-LU password, as specified in the definition of the password.

password_info.password
An encrypted version of the password supplied on a DEFINE_LU_LU_PASSWORD verb. This is an 8-byte hexadecimal string.

password_info.protocol_defined
Requested LU-LU verification protocol defined for use with this partner LU. Possible values are:

AP_BASIC
Basic security protocols requested.

AP_ENHANCED
Enhanced security protocols requested.

AP_EITHER
Basic or enhanced security accepted.

password_info.protocol_in_use
LU-LU verification protocol in use with this partner LU. Possible values are:

AP_BASIC
Basic security protocols in use.

AP_ENHANCED
Enhanced security protocols in use.

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

エイプ無効ルリエイリアス
指定された *lu_alias* パラメーターが、構成されたどの LU の別名とも一致しませんでした

ファイル名の変更
指定された *lu_name* パラメーターが、構成されたどの LU の名前とも一致しませんでした

ファイルの追加リスト・オプション
list_options パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

照会 LU_LU_POOL

QUERY_LU_POOL は、 LU プール、および LU プールに属する LU に関する情報を戻します。

この verb を使用して、使用するオプションに応じて、特定の LU またはプール、または複数の LU またはプールに関する情報を取得することができます。

VCB structure

```
typedef struct query_lu_pool
{
   AP_UINT16        opcode;                 /* verb operation code        */
   unsigned char    reserv2;                /* reserved                   */
   unsigned char    format;                 /* reserved                   */
   AP_UINT16        primary_rc;             /* primary return code        */
   AP_UINT32        secondary_rc;           /* secondary return code      */
   unsigned char    *buf_ptr;               /* pointer to buffer          */
   AP_UINT32        buf_size;               /* buffer size                */
   AP_UINT32        total_buf_size;         /* total buffer size required */
   AP_UINT16        num_entries;            /* number of entries          */
   AP_UINT16        total_num_entries;      /* total number of entries    */
   unsigned char    list_options;           /* listing options            */
   unsigned char    reserv3;                /* reserved                   */
   unsigned char    pool_name[8];           /* Pool name                  */
   unsigned char    lu_name[8];             /* LU name                    */
} QUERY_LU_POOL;
```

```
typedef struct lu_pool_summary
{
   AP_UINT16        overlay_size;           /* size of returned entry     */
   unsigned char    pool_name[8];           /* Pool name                  */
   unsigned char    description[32];        /* resource description       */
   unsigned char    reserv1[16];            /* reserved                   */
   AP_UINT16        num_active_lus;         /* number of active lus       */
   AP_UINT16        num_avail_lus;          /* number of available lus    */
} LU_POOL_SUMMARY;
```

```
typedef struct lu_pool_detail
{
   AP_UINT16        overlay_size;           /* size of returned entry     */
   unsigned char    pool_name[8];           /* Pool name                  */
   unsigned char    description[32];        /* resource description       */
   unsigned char    reserv1[16];            /* reserved                   */
```

```
    unsigned char    lu_name[8];                /* LU name                     */
    unsigned char    lu_sscp_sess_active;       /* Is LU-SSCP session active   */
    unsigned char    appl_conn_active;          /* Is appl connection open     */
    unsigned char    plu_sess_active;           /* Is PLU-SLU session active   */
} LU_POOL_DETAIL;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
追加の照会プール

buf_ptr
CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

num_entries
データが戻されるエントリーの最大数。 *list_options* が **要約の要約**に設定されている場合、各エントリーは単一の LU プールになります。 *list_options* が **追加の詳細**に設定されている場合は、各エントリーはプール内の LU (または空の LU プールを示すエントリー) になります。

特定の範囲ではなく特定の項目を要求するには、 1 という値を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、 CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータの戻りを開始する必要のあるリスト内の位置、および各項目に必要な情報のレベル。 以下の値のいずれかを使用して、情報のレベルを指定します。

要約の要約
要約情報のみ (LU プールのリスト) 。

追加の詳細
詳細情報 (LU プール内の個々の LU をリストします) 。

以下のいずれかの値を指定して、論理 **それとも** 操作を使用してこの値を結合します。

リストの最初のリスト （ _R ）
リストの最初の項目から開始します。

リストを含む （ 包括的 ）
プール名 パラメーターと *lu_name* パラメーターの組み合わせによって指定されたエントリーから開始します。

次への AP_LIST_FROM_NEXT
プール名 パラメーターと *lu_name* パラメーターの組み合わせによって指定されたエントリーの直後のエントリーから開始します。

リストは プール名 によって順序付けられ、次に *lu_name* によって順序付けられます。 リストの順序とアプリケーションでの特定のエントリーの取得方法について詳しくは、 34 ページの『List options for QUERY_* Verbs』を参照してください。

プール名
LU プールの名前。 *list_options* を **リストの最初のリスト** （ _R ）に設定すると、この値は無視されます。 これは 8 バイトの EBCDIC タイプ A ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

lu_name
LU 名。 *list_options* が **リストの最初のリスト** （ _R ）または **要約の要約**に設定されている場合、この値は無視されます これは 8 バイトのタイプ A の EBCDIC ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

プール内のすべての LU に関する情報を入手するには、 プール名 をプールの名前に設定し、 *num_entries* を 0 に設定し、 *lu_name* を 8 桁の 2 進ゼロに設定します。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

lu_pool_summary.overlay_size
The size of the returned `lu_pool_summary` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `lu_pool_summary` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

lu_pool_summary.pool_name
Name of LU pool. This is an 8-byte EBCDIC string, padded on the right with spaces if the name is shorter than 8 characters.

lu_pool_summary.description
A null-terminated text string describing the LU pool, as specified in the definition of the pool.

lu_pool_summary.num_active_lus
Number of LUs in the pool that are active.

lu_pool_summary.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.

lu_pool_detail.overlay_size
The size of the returned `lu_pool_detail` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `lu_pool_detail` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

lu_pool_detail.pool_name
Name of LU pool to which the LU belongs. This is an 8-byte EBCDIC string, padded on the right with spaces if the name is shorter than 8 characters.

lu_pool_detail.description
A null-terminated text string describing the LU pool, as specified in the definition of the pool.

lu_pool_detail.lu_name
LU name of the LU. This is an 8-byte type-A EBCDIC string, padded on the right with spaces if the name is shorter than 8 characters. If a single `lu_pool_detail` structure is returned for a particular pool name with a string of 8 binary zeros for the LU name, this indicates that the specified pool is empty.

lu_pool_detail.lu_sscp_sess_active
Specifies whether the LU-SSCP session is active. Possible values are:

AP_YES
The session is active.

AP_NO
The session is inactive.

lu_pool_detail.appl_conn_active
Specifies whether an application is using the LU. Possible values are:

AP_YES
An application is using the LU.

AP_NO
No application is using the LU.

lu_pool_detail.plu_sess_active
Specifies whether the PLU-SLU session is active. Possible values are:

AP_YES
The session is active.

AP_NO
The session is inactive.

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_LU_NAME
The *list_options* parameter was set to AP_LIST_INCLUSIVE to list all entries starting from the supplied name, but the *lu_name* parameter was not valid.

AP_INVALID_POOL_NAME
The *list_options* parameter was set to AP_LIST_INCLUSIVE to list all entries starting from the supplied name, but the *pool_name* parameter was not valid.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

QUERY_LU62_TIMEOUT

QUERY_LU62_TIMEOUT verb は、前に DEFINE_LU62_TIMEOUT verb で定義された LU タイプ 6.2 セッション・タイムアウトの定義に関する情報を戻します。

情報はリストとして戻されます。 特定のタイムアウト、またはいくつかのタイムアウト値に関する情報を取得するには、 リソース・タイプ パラメーターおよび リソース名 パラメーターの値を指定します。 *list_options* パラメーターが **リストの最初のリスト** （ _R ）に設定されている場合、 リソース・タイプ パラメーターおよび リソース名 パラメーターは無視されます。 返されるリストは、 リソース・タイプ 上で順序付けされ、その後 リソース名に配列され

リソース・タイプの場合、順序は次のとおりです。

1. グローバル・タイムアウト
2. ローカル LU タイムアウト
3. パートナー LU タイムアウト
4. モード・タイムアウト

リソース名の場合、順序付けは以下のようになります。

1. 名前長さ
2. 同じ長さの名前に対する ASCII 辞書順の順序付け

list_options パラメーターが **次への** AP_LIST_FROM_NEXT に設定されている場合、戻されたリストは、定義された順序 (指定されたエントリーが存在するかどうかに関係なく) に従って、次のエントリー用に開始されます。

VCB 構造体

```
typedef struct query_lu62_timeout
{
  AP_UINT16      opcode;                    /* verb operation code        */
  unsigned char  reserv2;
  unsigned char  format;                    /* reserved                   */
  AP_UINT16      primary_rc;                /* primary return code        */
  AP_UINT32      secondary_rc;              /* secondary return code      */
  unsigned char *buf_ptr;                   /* buffer pointer             */
  AP_UINT32      buf_size;                  /* buffer size                */
  AP_UINT32      total_buf_size;            /* total buffer size          */
  AP_UINT16      num_entries;               /* number of entries          */
  AP_UINT16      total_num_entries;         /* total number of entries    */
  unsigned char  list_options;              /* list options               */
  unsigned char  reserv3;                   /* reserved                   */
  unsigned char  resource_type;             /* resource type              */
  unsigned char  resource_name[17];         /* resource name              */
} QUERY_LU62_TIMEOUT;
```

```
typedef struct lu62_timeout_data
{
  AP_UINT16      overlay_size;              /* overlay size               */
  unsigned char  resource_type;             /* resource type              */
  unsigned char  resource_name[17];         /* resource name              */
  AP_UINT16      timeout;                   /* timeout                    */
} LU62_TIMEOUT_DATA;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_LU62_TIMEOUT

buf_ptr
A pointer to a data buffer that CS Linux will use to return the requested information.

buf_size
Size of the supplied data buffer.

num_entries
Maximum number of entries for which data should be returned. To request data for a specific entry rather than a range, specify the value 1. To return as many entries as possible, specify 0; in this case, CS Linux will return the maximum number of entries that can be accommodated in the supplied data buffer.

list_options
The position in the list of entries from which CS Linux begins to return data. The list is ordered by *resource_type* in the order AP_GLOBAL_TIMEOUT, AP_LOCAL_LU_TIMEOUT, AP_PARTNER_LU_TIMEOUT, AP_MODE_TIMEOUT, then by *resource_name* in order of the name length, then by ASCII lexicographical ordering for names of the same length.

Possible values are:

AP_FIRST_IN_LIST
Start at the first entry in the list

AP_LIST_INCLUSIVE
Start at the entry specified by the combination of the *resource_type* and *resource_name* parameters

AP_LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the combination of the *resource_type* and *resource_name* parameters

For more information about how the list is ordered and how the application can obtain specific entries from it, see "List options for QUERY_* Verbs" on page 34.

resource_type
Specifies the type of timeout being queried. This parameter is ignored if *list_options* is set to AP_FIRST_IN_LIST.

Possible values are:

AP_GLOBAL_TIMEOUT
Timeout applies to all LU 6.2 sessions for the local node.

AP_LOCAL_LU_TIMEOUT
Timeout applies to all LU 6.2 sessions for the local LU specified in the *resource_name* parameter.

AP_PARTNER_LU_TIMEOUT
Timeout applies to all LU 6.2 sessions to the partner LU specified in the *resource_name* parameter.

AP_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 AP_GLOBAL_TIMEOUT, do not specify this parameter.
- If *resource_type* is set to AP_LOCAL_LU_TIMEOUT, only the first 8 bytes of *resource_name* are valid and should be set to the name of the local LU. This is an 8-byte alphanumeric type-A EBCDIC string starting with a letter, padded to the right with EBCDIC spaces. Set the remaining nine bytes to all zeros.
- If *resource_type* is set to AP_PARTNER_LU_TIMEOUT, all 17 bytes of resource_name are valid and should be set to the fully-qualified name of the partner LU which is padded on the right with EBCDIC spaces. The name consists of a 1-8 type-A character network name, followed by an EBCDIC dot (period) character, followed by a 1-8 type-A character partner LU name.
- If *resource_type* is set to AP_MODE_TIMEOUT, only the first 8 bytes of resource_name are valid and should be set to the name of the mode. This is an 8-byte alphanumeric type-A EBCDIC string starting with a letter, padded to the right with EBCDIC spaces. Set he remaining 9 bytes to all zeros.

This parameter is ignored if *list_options* is set to AP_FIRST_IN_LIST.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

buf_size
提供されたバッファーに戻された情報の長さ。

total_buf_size
要求されたすべてのリスト情報を戻すために必要となるバッファーのサイズを示す戻り値。 *buf_size* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

num_entries
データ・バッファーに戻されたエントリーの数。

total_num_entries
使用可能な項目の合計数。 *num_entries* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

データ・バッファー内の各項目は、以下のパラメーターで構成されます。

リソース・タイプ
タイムアウトのタイプ。 可能な値は次のとおりです

グローバル・タイムアウトの追加
タイムアウトは、ローカル・ノードのすべての LU 6.2 セッションに適用されます。 リソース名 パラメーターは、すべてゼロに設定されます。

ローカル・タイムアウトの追加タイムアウト
タイムアウトは、 リソース名 パラメーターによって示されるローカル LU のすべての LU 6.2 セッションに適用されます。

パートナー LU_TIMEOUT の設定
タイムアウトは、 リソース名 パラメーターによって示されるパートナー LU へのすべての LU 6.2 セッションに適用されます。

AP_MODE_TIMEOUT
タイムアウトは、 リソース名 パラメーターで指示されたモードを使用するすべての LU 6.2 セッションに適用されます。

リソース名
リソースの名前。 この名前は、ローカル LU、パートナー LU、またはモードの 1 つで、 リソース・タイプ パラメーターの値によって決まります。 リソース・タイプ を **グローバル・タイムアウトの追加** に設定すると、このパラメーターはゼロに設定されます。

タイムアウト
タイムアウト期間 (秒)。 値 0 (ゼロ) は、セッションが解放された直後にタイムアウトになることを示します。

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_RESOURCE_NAME
The *list_options* parameter was set to AP_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.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

QUERY_MDS_APPLICATION

QUERY_MDS_APPLICATION 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 verb, see the *IBM Communications Server for Data Center Deployment on AIX or Linux MS Programmer's Guide*.

This verb can be used to obtain information about a specific application or about multiple applications, depending on the options used.

This verb must be issued to a running node.

VCB structure

```
typedef struct query_mds_application
{
   AP_UINT16        opcode;               /* verb operation code            */
   unsigned char    reserv2;              /* reserved                       */
   unsigned char    format;               /* reserved                       */
   AP_UINT16        primary_rc;           /* primary return code            */
   AP_UINT32        secondary_rc;         /* secondary return code          */
   unsigned char    *buf_ptr;             /* pointer to buffer              */
   AP_UINT32        buf_size;             /* buffer size                    */
   AP_UINT32        total_buf_size;       /* total buffer size required     */
   AP_UINT16        num_entries;          /* number of entries              */
   AP_UINT16        total_num_entries;    /* total number of entries        */
   unsigned char    list_options;         /* listing options                */
   unsigned char    reserv3;              /* reserved                       */
   unsigned char    application[8];       /* application                    */
} QUERY_MDS_APPLICATION;
```

```
typedef struct mds_application_data
{
   AP_UINT16        overlay_size;         /* size of returned entry         */
   unsigned char    application[8];       /* application name               */
   AP_UINT16        max_rcv_size;         /* max data size appl can receive */
   unsigned char    reserva[20];          /* reserved                       */
} MDS_APPLICATION_DATA;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_MDS_APPLICATION

buf_ptr
A pointer to a data buffer that CS Linux will use to return the requested information.

buf_size
Size of the supplied data buffer.

num_entries
Maximum number of applications for which data should be returned. To request data for a specific application rather than a range, specify the value 1. To return as many entries as possible, specify zero; in this case, CS Linux will return the maximum number of entries that can be accommodated in the supplied data buffer.

list_options
The position in the list of applications from which CS Linux should begin to return data. Possible values are:

AP_FIRST_IN_LIST
Start at the first entry in the list.

AP_LIST_INCLUSIVE
Start at the entry specified by the application parameter.

AP_LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the application parameter.

For more information about how the list is ordered and how the application can obtain specific entries from it, see “List options for QUERY_* Verbs” on page 34.

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 AP_FIRST_IN_LIST. This is an 8-byte type-A EBCDIC string, padded on the right with spaces if the name is shorter than 8 characters.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

buf_size
提供されたバッファーに戻された情報の長さ。

total_buf_size
要求されたすべてのリスト情報を戻すために必要となるバッファーのサイズを示す戻り値。 *buf_size* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

num_entries
データ・バッファーに戻されたエントリーの数。

total_num_entries
使用可能な項目の合計数。 *num_entries* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

データ・バッファー内の各項目は、以下のパラメーターで構成されます。

mds_application_data.オーバーレイ・サイズ
戻された mds_application_data 構造体のサイズ。すなわち、データ・バッファー内の次のエントリーの先頭までのオフセット。

アプリケーションが戻されたバッファーを調べて、各 mds_application_data 構造体を順番に検出する必要がある場合は、この値を使用して次のデータ構造の正しいオフセットに移動しなければなりません。また、 C sizeof() 演算子を使用してはなりません。 これは、返されるオーバーレイのサイズが、 CS Linux の将来のリリースで増加する可能性があるためです。戻されたオーバーレイ・サイズを使用すると、アプリケーションが今後のリリースで引き続き動作することが保証されます。

mds_application_data.application
登録済みアプリケーションの名前。 これは 8 バイトのタイプ A の EBCDIC ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

mds_application_data.max_rcv_size
アプリケーションが 1 つのメッセージで受信できる最大バイト数 (これは、アプリケーションが MDS に登録されたときに指定されます) 。 MDS レベル・アプリケーションの登録について詳しくは、「*AIX* または *Linux MS* プログラマーズ・ガイドでの *IBM Communications Server for Data Center* デプロイメント」を参照してください。

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

アプリケーション・アプリケーション名の追加
list_options パラメーターが **リストを含む** （ **包括的** ）に設定され、指定された名前から始まるすべての項目がリストされましたが、 申請 パラメーターが無効でした。

ファイルの追加リスト・オプション
list_options パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: function not supported

If the verb does not execute successfully because the local node configuration does not support it, CS Linux returns the following parameters:

primary_rc

AP_FUNCTION_NOT_SUPPORTED
The local node does not support MS network management functions; this is defined by the *mds_supported* parameter on the DEFINE_NODE verb.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

QUERY_MDS_STATISTICS

QUERY_MDS_STATISTICS returns Management Services statistics. This verb 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 using the DEFINE_NODE verb.

This verb must be issued to a running node.

VCB structure

```
typedef struct query_mds_statistics
{
   AP_UINT16       opcode;                   /* verb operation code         */
   unsigned char   reserv2;                  /* reserved                    */
   unsigned char   format;                   /* reserved                    */
   AP_UINT16       primary_rc;               /* primary return code         */
   AP_UINT32       secondary_rc;             /* secondary return code       */
   AP_UINT32       alerts_sent;              /* number of alert sends       */
   AP_UINT32       alert_errors_rcvd;        /* error messages received     */
                                             /*  for alert sends            */
   AP_UINT32       uncorrelated_alert_errors; /* uncorrelated alert errors  */
                                             /*  received                   */
   AP_UINT32       mds_mus_rcvd_local;       /* number of MDS_MUs received  */
                                             /*  from local applications    */
   AP_UINT32       mds_mus_rcvd_remote;      /* number of MDS_MUs received  */
                                             /*  from remote applications   */
   AP_UINT32       mds_mus_delivered_local;  /* number of MDS_MUs delivered */
                                             /*  to local applications      */
   AP_UINT32       mds_mus_delivered_remote; /* number of MDS_MUs delivered */
                                             /*  to remote applications     */
   AP_UINT32       parse_errors;             /* number of MDS_MUs received  */
                                             /*  with parse errors          */
   AP_UINT32       failed_deliveries;        /* number of MDS_MUs where     */
```

```
                                          /*  delivery failed            */
   AP_UINT32       ds_searches_performed;  /* number of DS searches       */
                                          /*  performed                  */
   AP_UINT32       unverified_errors;      /* number of unverified errors */
   unsigned char   reserva[20];            /* reserved                    */
} QUERY_MDS_STATISTICS;
```

Supplied parameters

The application supplies the following parameter:

opcode
AP_QUERY_MDS_STATISTICS

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

alerts_sent
Number of locally originated alerts sent using the MDS transport system.

alert_errors_rcvd
Number of error messages received by MDS indicating a delivery failure for a message containing an alert.

uncorrelated_errors_rcvd
Number of error messages received by MDS indicating a delivery failure for a message containing an alert. Delivery failure occurs when the error message could not be correlated to an alert on the MDS send alert queue. MDS maintains a fixed-size queue where it caches alerts sent to the problem determination focal point. Once the queue reaches maximum size, the oldest alert will be 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 that the alert may be held until the problem determination focal point is restored.

Note : The two counts *alert_errors_rcvd* and *uncorrelated_errors_rcvd* can be used to check that the size of the send alert queue (specified on DEFINE_NODE) is appropriate. If the value of *uncorrelated_errors_rcvd* increases over time, this indicates that the send alert queue size 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 a remote node using the MDS_SEND transaction program.

parse_errors
Number of MDS_MUs received which contained header format errors.

failed_deliveries
Number of MDS_MUs this node failed to deliver.

ds_searches_performed
Number of Directory Services searches used to locate the next hop for an MDS_MU. (Significant for network nodes only).

unverified_errors
Number of routing errors due to using unverified (local Directory Services search) data for determining the next hop for an MDS_MU. Each time one of these errors occurs, Directory Services must repeat the search using either a Central Directory Search or a broadcast search mechanism. (Significant for network nodes only).

戻りパラメーター: 関数はサポートされません

ローカル・ノード構成がそれをサポートしていないために verb が正常に実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc

付加機能がサポートされていません
ローカル・ノードは MS ネットワーク管理機能をサポートしていません。これは、 DEFINE_NODE verb の *mds_supported* パラメーターによって定義されます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

QUERY_MODE

QUERY_MODE returns information about modes that a local LU is using, or has used, with partner LUs.

This verb can be used to obtain information about a specific partner LU-mode combination or about multiple modes, and about modes for which sessions are currently active or about all modes that have been used, depending on the options used. This verb returns information about current usage of the modes and LUs, not about their definition; use QUERY_MODE_DEFINITION to obtain the definition of the modes and LUs.

This verb must be issued to a running node.

VCB structure

```
typedef struct query_mode
{
   AP_UINT16       opcode;               /* verb operation code             */
   unsigned char   reserv2;              /* reserved                        */
   unsigned char   format;               /* reserved                        */
   AP_UINT16       primary_rc;           /* primary return code             */
   AP_UINT32       secondary_rc;         /* secondary return code           */
   unsigned char   *buf_ptr;             /* pointer to buffer               */
   AP_UINT32       buf_size;             /* buffer size                     */
   AP_UINT32       total_buf_size;       /* total buffer size required      */
   AP_UINT16       num_entries;          /* number of entries               */
   AP_UINT16       total_num_entries;    /* total number of entries         */
   unsigned char   list_options;         /* listing options                 */
   unsigned char   reserv3;              /* reserved                        */
   unsigned char   lu_name[8];           /* LU name                         */
   unsigned char   lu_alias[8];          /* LU alias                        */
   unsigned char   plu_alias[8];         /* partner LU alias                */
   unsigned char   fqplu_name[17];       /* fully qualified partner LU name */
   unsigned char   mode_name[8];         /* mode name                       */
   unsigned char   active_sessions;      /* active sessions only filter     */
} QUERY_MODE;
```

```
typedef struct mode_summary
{
   AP_UINT16       overlay_size;         /* size of returned entry          */
   unsigned char   mode_name[8];         /* mode name                       */
   unsigned char   description[32];      /* resource description            */
   unsigned char   reserv2[16];          /* reserved                        */
   AP_UINT16       sess_limit;           /* current session limit           */
   AP_UINT16       act_sess_count;       /* currently active sessions count */
   unsigned char   fqplu_name[17];       /* fully-qualified partner LU name */
```

```
   unsigned char   reserv1[3];           /* reserved                       */
} MODE_SUMMARY;
```

```
typedef struct mode_detail
{
   AP_UINT16       overlay_size;          /* size of returned entry         */
   unsigned char   mode_name[8];          /* mode name                      */
   unsigned char   description[32];       /* resource description           */
   unsigned char   reserv2[16];           /* reserved                       */
   AP_UINT16       sess_limit;            /* session limit                  */
   AP_UINT16       act_sess_count;        /* currently active sessions count */
   unsigned char   fqplu_name[17];        /* fully-qualified partner LU name */
   unsigned char   reserv1[3];            /* reserved                       */
   AP_UINT16       min_conwinners_source; /* minimum conwinner sess limit   */
   AP_UINT16       min_conwinners_target; /* minimum conloser sess limit    */
   unsigned char   drain_source;          /* drain source?                  */
   unsigned char   drain_partner;         /* drain partner?                 */
   AP_UINT16       auto_act;              /* auto activated conwinner       */
                                          /*  session limit                 */
   AP_UINT16       act_cw_count;          /* active conwinner sessions count */
   AP_UINT16       act_cl_count;          /* active conloser sessions count  */
   unsigned char   sync_level;            /* synchronization level          */
   unsigned char   default_ru_size;       /* default RU size to maximize    */
                                          /*  performance                   */
   AP_UINT16       max_neg_sess_limit;    /* maximum negotiated session limit*/
   AP_UINT16       max_rcv_ru_size;       /* maximum receive RU size        */
   AP_UINT16       pending_session_count; /* pending sess count for mode    */
   AP_UINT16       termination_count;     /* termination count for mode     */
   AP_UINT16       implicit;              /* implicit or explicit entry     */
   unsigned char   reserva[15];           /* reserved                       */
} MODE_DETAIL;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_MODE

buf_ptr
A pointer to a data buffer that CS Linux will use to return the requested information.

buf_size
Size of the supplied data buffer.

num_entries
Maximum number of modes for which data should be returned. To request data for a specific mode rather than a range, specify the value 1. To return as many entries as possible, specify zero; in this case, CS Linux will return the maximum number of entries that can be accommodated in the supplied data buffer.

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

AP_SUMMARY
Summary information only.

AP_DETAIL
Detailed information.

Combine this value using a logical OR operation with one of the following values:

AP_FIRST_IN_LIST
Start at the first entry in the list (the first partner LU and mode for the specified local LU).

AP_LIST_INCLUSIVE
Start at the entry specified by the supplied partner LU name and mode name.

AP_LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the supplied partner LU name and mode name.

For AP_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 AP_LIST_INCLUSIVE or AP_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. For more information about how the list is ordered and how the application can obtain specific entries from it, see “List options for QUERY_* Verbs” on page 34.

lu_name
LU name. This is an 8-byte type-A EBCDIC string, padded on the right with spaces if the name is shorter than 8 characters. To specify that the LU is identified by its alias rather than its LU name, set this parameter to 8 binary zeros and specify the LU alias in the following parameter.

lu_alias
Locally defined LU alias. This parameter is used only if *lu_name* is set to 8 binary zeros; it is ignored otherwise.

The alias is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes. To indicate the LU associated with the CP (the default LU), set both *lu_name* and *lu_alias* to binary zeros.

plu_alias
Partner LU alias. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes. If *list_options* is set to AP_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 specify that the LU is identified by its LU name rather than its alias, set this parameter to 8 binary zeros and specify the LU name in the following parameter.

fqplu_name
Fully qualified network name for the partner LU. If *list_options* is set to AP_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 *plu_alias* is set to 8 binary zeros; it is ignored otherwise.

The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

mode_name
Mode name which designates the network properties for a group of sessions. This is an 8-byte type-A EBCDIC string, padded on the right with spaces if the name is shorter than 8 characters. This parameter is ignored if *list_options* is set to AP_FIRST_IN_LIST.

active_sessions
Specifies whether to return information only on modes for which sessions are active, or on all modes. Possible values are:

AP_YES
Return information only on modes for which sessions are currently active.

AP_NO
Return information about all modes for which sessions are active or have been active.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

buf_size
提供されたバッファーに戻された情報の長さ。

total_buf_size
要求されたすべてのリスト情報を戻すために必要となるバッファーのサイズを示す戻り値。 *buf_size* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

num_entries
データ・バッファーに戻されたエントリーの数。

total_num_entries
使用可能な項目の合計数。 *num_entries* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

データ・バッファー内の各項目は、以下のパラメーターで構成されます。

mode_summary.オーバーレイ・サイズ
戻された **モードの要約** 構造体のサイズ。すなわち、データ・バッファー内の次のエントリーの先頭までのオフセット。

アプリケーションが戻されたバッファーを調べて、各 **モードの要約** 構造体を順番に検出する必要がある場合は、この値を使用して次のデータ構造の正しいオフセットに移動しなければなりません。また、 C `sizeof()` 演算子を使用してはなりません。 これは、返されるオーバーレイのサイズが 、 CS Linux の将来のリリースで増加する可能性があるためです。戻されたオーバーレイ・サイズを使用すると、アプリケーションが今後のリリースで引き続き動作することが保証されます。

mode_summary.mode_name
モード名。 これは 8 バイトのタイプ A の EBCDIC ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

mode_summary.description
モードの定義に指定されている、モードを記述するヌル終了のテキスト・ストリング。

mode_summary.sess_limit
現行セッション限度。

mode_summary.act_sess_count
モードを使用する、指定されたローカル LU とパートナー LU との間のアクティブ・セッションの総数。

mode_summary.fqplu_name
パートナー LU の完全修飾名。 この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

mode_detail.オーバーレイサイズ
戻された `mode_detail` 構造体のサイズ。すなわち、データ・バッファー内の次のエントリーの先頭までのオフセット。

アプリケーションが戻されたバッファーを調べて、各 `mode_detail` 構造体を順番に検出する必要がある場合は、この値を使用して次のデータ構造の正しいオフセットに移動しなければなりません。また、 C `sizeof()` 演算子を使用してはなりません。 これは、返されるオーバーレイのサイズが 、 CS Linux の将来のリリースで増加する可能性があるためです。戻されたオーバーレイ・サイズを使用すると、アプリケーションが今後のリリースで引き続き動作することが保証されます。

mode_detail.mode_name
モード名。 これは 8 バイトのタイプ A の EBCDIC ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

mode_detail.description
モードの定義に指定されている、モードを記述するヌル終了のテキスト・ストリング。

mode_detail.sess_limit
現行セッション限度。

mode_detail.act_sess_count
モードを使用する、指定されたローカル LU とパートナー LU との間のアクティブ・セッションの総数。

mode_detail.fqplu_name
パートナー LU の完全修飾名。 この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

mode_detail.min_conwinners_source
ローカル LU がコンテンション勝者となるセッションの最小数を指定します。

mode_detail.min_conwinners_target
ローカル LU がコンテンション敗者であるセッションの最小数を指定します。

mode_detail.drain_source
セッション限度が変更またはリセットされたときに、セッションを非活動化する前に、ローカル LU が待機セッション要求を満たすかどうかを指定 可能な値は次のとおりです

類人猿
セッションが非活動化される前に、待機セッション要求が満たされる

アプ・ノー
待機中のセッション要求は満たされません。

mode_detail.drain_パートナー
セッション限度が変更またはリセットされたときに、セッションを非活動化する前に、パートナー LU が待機セッション要求を満たすかどうかを指定 可能な値は次のとおりです

類人猿
セッションが非活動化される前に、待機セッション要求が満たされる

アプ・ノー
待機中のセッション要求は満たされません。

mode_detail.auto_act
パートナー LU との CNOS 交換の後で自動的に活動化される競合勝者セッションの数。

mode_detail.act_cw_count
このモードを使用しているアクティブな競合勝者セッションの数。(これらのセッションのいずれかを使用する前に、ローカル LU は "入札" を必要としません。

mode_detail.act_cl_count
このモードを使用するアクティブなコンテンション敗者セッションの数。(ローカル LU は、これらのセッションのいずれかを使用する前に "入札" を指定する

mode_detail.sync_level
モードによってサポートされる同期レベルを指定します。 可能な値は次のとおりです

確認の確認
このモードは、 CONFIRM verb および CONFIRMED verb を使用して同期をサポートします。

同期化の実行
モードは、同期点機能をサポートします。

追加なし
このモードは同期をサポートしません。

mode_detail.default_ru_size
最大 RU サイズのデフォルトの 上限を使用するかどうかを指定します。 可能な値は次のとおりです

類人猿
CS Linux は、モードの定義に指定されている最大 RU サイズを無視し、最大 RU サイズの 上限を、リンク BTU サイズに収容できる最大値に設定します。

アプ・ノー
CS Linux は、モードの定義に指定されている最大 RU サイズを使用します。

mode_detail.max_禰_sess_limit
交渉可能な最大セッション限度。 ローカル LU が、その CNOS 処理中にターゲット LU としてこのモード名で使用できる最大セッション限度を指定します。

mode_detail.max_rcv_ru_size
受信した最大 RU サイズ。

mode_detail.pending_session_count
保留中のセッションの数を指定します (セッションの活動化の完了を待機中) 。

mode_detail.termination_count
以前の CNOS verb でモード・セッション限度がゼロに設定されていて、会話がそれらを使用しているか、それらを使用するのを待っているためにセッションがまだアクティブである場合、このパラメーターは、まだ非活動化されていないセッションの数を指定します。

mode_detail.implicit
項目が暗黙定義または明示定義によって作成されたかどうかを示します。 可能な値は次のとおりです

類人猿
この項目は暗黙項目です。

アプ・ノー
エントリーは明示的なエントリーです。

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

ファイルの追加リスト・オプション
list_options パラメーターが有効な値に設定されていませんでした。

エイプ無効ルリエイリアス
list_options パラメーターが **リストを含む** （ **包括的** ）に設定され、指定された名前から始まるすべての項目がリストされましたが、 *lu_alias* パラメーターが無効でした。

ファイル名の変更
list_options パラメーターが **リストを含む** （ **包括的** ）に設定され、指定された名前から始まるすべての項目がリストされましたが、 *lu_name* パラメーターが無効でした。

ファイル・パス名
list_options パラメーターが **リストを含む** （ **包括的** ）に設定され、指定された名前から始まるすべての項目がリストされましたが、 モード名 パラメーターが無効でした。

ファイル名の変更
list_options パラメーターが **リストを含む** （ **包括的** ）に設定され、指定された名前から始まるすべての項目がリストされましたが、以下のいずれかの条件が適用されます。

- *fqplu_name* パラメーターが、このローカル LU のいずれかのパートナーの名前と一致しません。
- ローカル LU、パートナー LU、およびモードの指定された組み合わせに対して、(ノードが最後に開始されてから) アクティブなセッションがありませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

照会モードの定義

QUERY_MODE_DEFINITION は、 DEFINE_MODE を使用して定義されたモード、または SNA 定義のモードに関する情報を戻します。

この verb は、使用されるオプションに応じて、特定のモードまたは複数のモードに関する要約情報または詳細情報のいずれかを取得するために使用できます。 この関数は、モードの定義に関する情報を、現在の使用法についてではなく戻します。 QUERY_MODE を使用して、ローカルおよびパートナー LU によるモードの現在の使用状況に関する情報を取得します。

この verb を使用して、認識されないモード名に使用されるデフォルトの COS 名に関する情報を戻すことはできません。これを行うには、 QUERY_MODE_TO_COS_MAPPING を使用します。

VCB 構造体

```
typedef struct query_mode_definition
{
   AP_UINT16         opcode;                 /* verb operation code        */
   unsigned char     reserv2;                /* reserved                   */
   unsigned char     format;                 /* reserved                   */
   AP_UINT16         primary_rc;             /* primary return code        */
   AP_UINT32         secondary_rc;           /* secondary return code      */
   unsigned char     *buf_ptr;               /* pointer to buffer          */
   AP_UINT32         buf_size;               /* buffer size                */
   AP_UINT32         total_buf_size;         /* total buffer size required */
   AP_UINT16         num_entries;            /* number of entries          */
   AP_UINT16         total_num_entries;      /* total number of entries    */
   unsigned char     list_options;           /* listing options            */
   unsigned char     reserv3;                /* reserved                   */
   unsigned char     mode_name[8];           /* mode name                  */
} QUERY_MODE_DEFINITION;
```

```
typedef struct mode_def_summary
{
   AP_UINT16         overlay_size;           /* size of returned entry     */
   unsigned char     mode_name[8];           /* mode name                  */
   unsigned char     description[32];        /* resource description       */
   unsigned char     reserv1[16];            /* reserved                   */
} MODE_DEF_SUMMARY;
```

```
typedef struct mode_def_detail
{
   AP_UINT16         overlay_size;           /* size of returned entry     */
   unsigned char     mode_name[8];           /* mode name                  */
   MODE_CHARS        mode_chars;             /* mode characteristics       */
} MODE_DEF_DETAIL;
```

```
typedef struct mode_chars
{
   unsigned char     description[32];         /* resource description       */
   unsigned char     reserv2[16];            /* reserved                   */
   AP_UINT16         max_ru_size_upp;        /* maximum RU size upper bound*/
   unsigned char     receive_pacing_win;     /* receive pacing window      */
   unsigned char     default_ru_size;        /* default RU size to         */
                                             /*  maximize performance      */
   AP_UINT16         max_neg_sess_lim;       /* maximum negotiable session */
                                             /*  limit                     */
   AP_UINT16         plu_mode_session_limit; /* LU-mode session limit      */
   AP_UINT16         min_conwin_src;         /* minimum source contention  */
                                             /*  winner sessions           */
   unsigned char     cos_name[8];            /* class of service name      */
   unsigned char     cryptography;           /* cryptography (reserved)    */
   unsigned char   compression;             /* data compression supported? */
   AP_UINT16         auto_act;               /* number of sessions to be   */
                                             /* activated automatically    */
   AP_UINT16         min_conloser_src;       /* minimum source contention  */
                                             /*  loser                     */
   AP_UINT16         max_ru_size_low;        /* maximum RU size lower bound*/
   AP_UINT16         max_receive_pacing_win; /* maximum receive pacing     */
                                             /*  window                    */
   unsigned char   max_compress_lvl;      /* max level of data compression   */
   unsigned char   max_decompress_lvl;    /* max level of data decompression */
   unsigned char     comp_in_series;         /* reserved                   */
   unsigned char     reserv4[25];            /* reserved                   */
} MODE_CHARS;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
AP_QUERY_MODE_DEFINITION

buf_ptr
CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

num_entries
データが戻されるモードの最大数。 特定の範囲ではなく、特定のモードのデータを要求するには、 1 という値を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、 CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータの戻りを開始する必要のあるリスト内の位置、および各項目に必要な情報のレベル。 以下の値のいずれかを使用して、情報のレベルを指定します。

要約の要約
サマリー情報のみ。

追加の詳細
詳細情報。

以下のいずれかの値を指定して、論理 **それとも** 操作を使用してこの値を結合します。

リストの最初のリスト （ _R ）
リストの最初の項目から開始します。

リストを含む （ 包括的 ）
モード名 パラメーターで指定されたエントリーから開始します。

次への AP_LIST_FROM_NEXT
モード名 パラメーターによって指定されたエントリーの直後のエントリーから開始します。

アプリケーションがリストから特定の項目を取得する方法について詳しくは、34 ページの『List options for QUERY_* Verbs』を参照してください。 この verb は、モードが作成順にリストされているという点で、他の QUERY_* verb とは異なります。

モード名
セッション・グループのネットワーク・プロパティーを指定するモード名。 *list_options* を **リストの最初のリスト** （ _R ）に設定すると、このパラメーターは無視されます。 これは 8 バイトのタイプ A の EBCDIC ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

mode_def_summary.overlay_size
The size of the returned `mode_def_summary` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `mode_def_summary` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

mode_def_summary.mode_name
Mode name. This is an 8-byte type-A EBCDIC string, padded on the right with spaces if the name is shorter than 8 characters.

mode_def_summary.description
A null-terminated text string describing the mode, as specified in the definition of the mode.

mode_def_detail.overlay_size
The size of the returned `mode_def_detail` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `mode_def_detail` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

mode_def_detail.mode_name
Mode name. This is an 8-byte type-A EBCDIC string, padded on the right with spaces if the name is shorter than 8 characters.

mode_def_detail.mode_chars.description
A null-terminated text string describing the mode, as specified in the definition of the mode.

mode_def_detail.mode_chars.max_ru_size_upp
Upper boundary for the maximum RU size to be used on sessions with this mode name. The value is used when the maximum RU size is negotiated during session activation.

Range: 256-61,440. This field is ignored if the *default_ru_size* parameter (see below) is set to AP_YES.

mode_def_detail.mode_chars.receive_pacing_win
Session pacing window for sessions using this mode. For fixed pacing, this is the maximum number of frames that can be received from the partner LU before the local LU must send a response; for adaptive pacing, this value is used as an initial receive window size. CS Linux always uses adaptive pacing unless the adjacent node specifies that it is not supported.

Range is 1-63, or zero to specify no pacing window (that is, an unlimited number of frames can be received, and no response is required).

mode_def_detail.mode_chars.default_ru_size
Specifies whether a default upper bound for the maximum RU size will be used. Possible values are:

AP_YES
CS Linux ignores the *max_ru_size_upp* parameter, and sets the upper bound for the maximum RU size to the largest value that can be accommodated in the link BTU size.

AP_NO
CS Linux uses the *max_ru_size_upp* parameter to define the maximum RU size.

mode_def_detail.mode_chars.max_neg_sess_lim
Maximum number of sessions allowed on this mode between any local LU and partner LU. Range: 1-32,767, or zero to specify no implicit CNOS exchange.

mode_def_detail.mode_chars.plu_mode_session_limit
Default session limit for this mode. This 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. Range: 1-32,767, or zero to specify no implicit CNOS exchange.

mode_def_detail.mode_chars.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 implicitly. Range: 1-32,767, or zero to specify no implicit CNOS exchange.

mode_def_detail.mode_chars.cos_name
Name of the class of service to request when activating sessions on this mode. This is an 8-byte type-A EBCDIC string, padded on the right with spaces if the name is shorter than 8 characters.

mode_def_detail.mode_chars.compression
Specifies whether sessions activated using this mode can use compression. Possible values are:

AP_COMP_PROHIBITED
Compression is not supported for sessions using this mode.

AP_COMP_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.)

mode_def_detail.mode_chars.auto_act
Specifies how many sessions will be activated automatically for this mode. This value is used when CNOS (Change Number of Sessions) exchange is initiated implicitly. This value is in the range 0-32,767.

mode_def_detail.mode_chars.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. This value is in the range 0-32,767.

mode_def_detail.mode_chars.max_ru_size_low
Lower bound for the maximum size of RUs sent and received on sessions that use this mode.

This value is in the range 256-61,440 or zero, which means that there is no lower bound.

mode_def_detail.mode_chars.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. (CS Linux always uses adaptive pacing unless the adjacent node specifies that it does not support it.)

This value is in the range 0-32,767 or zero, which means there is no limit for the pacing window.

mode_def_detail.mode_chars.max_compress_lvl
Specifies the maximum level of compression that CS Linux will attempt to negotiate for data flowing from the local node. Possible values are:

- AP_NONE
- AP_RLE_COMPRESSION
- AP_LZ9_COMPRESSION
- AP_LZ10_COMPRESSION

If compression is negotiated using a non-extended BIND, which does not specify a maximum compression level, RLE compression will be used.

mode_def_detail.mode_chars.max_decompress_lvl
Specifies the maximum level of decompression that CS Linux will attempt to negotiate for data flowing into the local node. Possible values are:

- AP_NONE
- AP_RLE_COMPRESSION
- AP_LZ9_COMPRESSION
- AP_LZ10_COMPRESSION

If compression is negotiated using a non-extended BIND, which does not specify a maximum compression level, RLE compression will be used.

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

ファイル・ パス名
list_options パラメーターが **リストを含む** (**包括的**) に設定され、指定された名前から始まるすべての項目がリストされましたが、 モード名 パラメーターが無効でした。

ファイルの追加リスト・ オプション
list_options パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

クエリー・モードからのマッピング (マッピング)

QUERY_MODE_TO_COS_MAPPING は、特定のモードに関連付けられている COS (サービス・クラス) に関する情報を戻します。 この verb は、使用されるオプションに応じて、特定のモードまたは複数のモードに関する情報を取得するために使用できます。

この verb は実行中のノードに対して発行する必要があります

VCB 構造体

```
typedef struct query_mode_to_cos_mapping
{
   AP_UINT16        opcode;                 /* verb operation code         */
   unsigned char    reserv2;                /* reserved                    */
   unsigned char    format;                 /* reserved                    */
   AP_UINT16        primary_rc;             /* primary return code         */
   AP_UINT32        secondary_rc;           /* secondary return code       */
   unsigned char    *buf_ptr;               /* pointer to buffer           */
   AP_UINT32        buf_size;               /* buffer size                 */
   AP_UINT32        total_buf_size;         /* total buffer size required  */
   AP_UINT16        num_entries;            /* number of entries           */
   AP_UINT16        total_num_entries;      /* total number of entries     */
   unsigned char    list_options;           /* listing options             */
   unsigned char    reserv3;                /* reserved                    */
   unsigned char    mode_name[8];           /* mode name                   */
} QUERY_MODE_TO_COS_MAPPING;
```

```
typedef struct mode_to_cos_mapping_data
{
   AP_UINT16        overlay_size;           /* size of returned entry      */
   unsigned char    mode_name[8];           /* mode name                   */
   unsigned char    cos_name[8];            /* cos name                    */
   unsigned char    reserva[20];            /* reserved                    */
} MODE_TO_COS_MAPPING_DATA;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
AP_QUERY_MODE_TO_COS_MAPPING

buf_ptr
CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

num_entries
データが戻されるモードの最大数。 特定の範囲ではなく、特定のモードのデータを要求するには、 1 という値を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、 CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータの戻りを開始するモードのリスト内の位置。 可能な値は次のとおりです

リストの最初のリスト （ _R ）
リストの最初の項目から開始します。

リストを含む （ 包括的 ）
モード名 パラメーターで指定されたエントリーから開始します。

次への AP_LIST_FROM_NEXT
モード名 パラメーターによって指定されたエントリーの直後のエントリーから開始します。

リストの順序とアプリケーションでの特定のエントリーの取得方法について詳しくは、 34 ページの『List options for QUERY_* Verbs』を参照してください。

モード名
情報が必要なモード名、またはリスト内の索引として使用される名前。 *list_options* を **リストの最初のリスト** （ _R ）に設定すると、この値は無視されます。

モード名は 8 バイトのタイプ A の EBCDIC ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。 認識されないモード名に使用されるデフォルトの COS に関する情報を戻すには、このパラメーターを 8 桁の 2 進ゼロに設定します。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

buf_size
提供されたバッファーに戻された情報の長さ。

total_buf_size
要求されたすべてのリスト情報を戻すために必要となるバッファーのサイズを示す戻り値。 *buf_size* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

num_entries
データ・バッファーに戻されたエントリーの数。

total_num_entries
使用可能な項目の合計数。 *num_entries* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

データ・バッファー内の各項目は、以下のパラメーターで構成されます。

***mode_to_cos_mapping_data.*オーバーレイのサイズ**
戻された mode_to_cos_mapping_data 構造体のサイズ。すなわち、データ・バッファー内の次のエントリーの先頭までのオフセット。

アプリケーションが戻されたバッファーを調べて、各 mode_to_cos_mapping_data 構造体を順番に検出する必要がある場合は、この値を使用して次のデータ構造の正しいオフセットに移動しなければなりません。また、 C sizeof() 演算子を使用してはなりません。 これは、返されるオーバーレイのサイズが、 CS Linux の将来のリリースで増加する可能性があるためです。戻されたオーバーレイ・サイズを使用すると、アプリケーションが今後のリリースで引き続き動作することが保証されます。

mode_to_cos_mapping_data.mode_name
モード名。 これは 8 バイトのタイプ A の EBCDIC ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

***mode_to_cos_mapping_data.cos_name* のモード名**
モード名に関連付けられているサービス・クラス名。 これは 8 バイトのタイプ A の EBCDIC ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_MODE_NAME
The *list_options* parameter was set to AP_LIST_INCLUSIVE to list all entries starting from the supplied name, but the *mode_name* parameter was not valid.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

QUERY_NMVT_APPLICATION

QUERY_NMVT_APPLICATION returns a list of applications that have registered for NMVT-level messages by issuing the MS verb REGISTER_NMVT_APPLICATION. For more information about this verb, see the *IBM Communications Server for Data Center Deployment on AIX or Linux MS Programmer's Guide*.

This verb can be used to obtain information about a specific application or about multiple applications, depending on the options used.

This verb must be issued to a running node.

VCB 構造体

```
typedef struct query_nmvt_application
{
   AP_UINT16         opcode;                 /* Verb operation code       */
   unsigned char     reserv2;                /* reserved                  */
   unsigned char     format;                 /* reserved                  */
   AP_UINT16         primary_rc;             /* primary return code       */
   AP_UINT32         secondary_rc;           /* secondary return code     */
   unsigned char     *buf_ptr;               /* pointer to buffer         */
   AP_UINT32         buf_size;               /* buffer size               */
   AP_UINT32         total_buf_size;         /* total buffer size required*/
   AP_UINT16         num_entries;            /* number of entries         */
   AP_UINT16         total_num_entries;      /* total number of entries   */
   unsigned char     list_options;           /* listing options           */
   unsigned char     reserv3;                /* reserved                  */
   unsigned char     application[8];         /* application               */
} QUERY_NMVT_APPLICATION;
```

```
typedef struct nmvt_application_data
{
   AP_UINT16         overlay_size;           /* size of returned entry    */
   unsigned char     application[8];         /* application name          */
   AP_UINT16         ms_vector_key_type;     /* MS vector key accepted    */
```

```
                                                /*  by appl                  */
    unsigned char     conversion_required;      /* is conversion to MDS_MU   */
                                                /*  required                 */
    unsigned char     reserv[5];                /* reserved                  */
    unsigned char     reserva[20];              /* reserved                  */
} NMVT_APPLICATION_DATA;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
AP_QUERY_NMVT_アプリケーション

buf_ptr
CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

num_entries
データが戻されるアプリケーションの最大数。 特定の範囲ではなく、特定のアプリケーションのデータを要求するには、 1 という値を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、 CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータの戻りを開始する必要のあるアプリケーションのリスト内の位置。 可能な値は次のとおりです

リストの最初のリスト （ _R ）
リストの最初の項目から開始します。

リストを含む （ 包括的 ）
アプリケーション・パラメーターで指定されたエントリーから開始します。

次への AP_LIST_FROM_NEXT
アプリケーション・パラメーターによって指定されたエントリーの直後のエントリーから開始します。

リストの順序とアプリケーションでの特定のエントリーの取得方法について詳しくは、 34 ページの『List options for QUERY_* Verbs』を参照してください。

申請
アプリケーション名。 *list_options* を **リストの最初のリスト （ _R ）**に設定すると、このパラメーターは無視されます。 この名前は 8 バイトのタイプ A の EBCDIC ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

buf_size
提供されたバッファーに戻された情報の長さ。

total_buf_size
要求されたすべてのリスト情報を戻すために必要となるバッファーのサイズを示す戻り値。 *buf_size* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

num_entries
データ・バッファーに戻されたエントリーの数。

total_num_entries
使用可能な項目の合計数。 *num_entries* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

データ・バッファー内の各項目は、以下のパラメーターで構成されます。

nmvt_application_data.オーバーレイ ・サイズ
戻された nmvt_application_data 構造体のサイズ。すなわち、データ・バッファー内の次のエントリーの先頭までのオフセット。

アプリケーションが戻されたバッファーを調べて、各 nmvt_application_data 構造体を順番に検出する必要がある場合は、この値を使用して次のデータ構造の正しいオフセットに移動しなければなりません。また、 C sizeof() 演算子を使用してはなりません。 これは、返されるオーバーレイのサイズが、 CS Linux の将来のリリースで増加する可能性があるためです。戻されたオーバーレイ ・サイズを使用すると、アプリケーションが今後のリリースで引き続き動作することが保証されます。

nmvt_application_data.application
登録済みアプリケーションの名前。 これは 8 バイトのタイプ A の EBCDIC ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

nmvt_application_data.ms_vector_key_type
アプリケーションによって受け入れられる MS ベクトル・キー。 アプリケーションは、 NMVT メッセージに登録するときに、どの MS ベクトル・キーを受け入れるかを指定します。

nmvt_application_data.conversion_required
登録済みアプリケーションが、着信メッセージを NMVT から MDS_MU フォーマットに変換する必要があるかどうかを指定します。 アプリケーションが NMVT メッセージに登録するときに、この変換が必要かどうかを指定します。 可能な値は次のとおりです

類人猿
着信メッセージは MDS_MU フォーマットに変換されます。

アプ・ノー
着信メッセージは変換されません。

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_APPLICATION_NAME
The *list_options* parameter was set to AP_LIST_INCLUSIVE to list all entries starting from the supplied name, but the application parameter was not valid.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

QUERY_NN_TOPOLOGY_NODE

Each network node maintains a network topology database which holds information about all the network nodes, virtual routing nodes (VRNs), and network node to network node TGs in the network. QUERY_NN_TOPOLOGY_NODE returns information about the network node and VRN entries in this database.

This verb can be used to obtain either summary or detailed information, about a specific node or about multiple nodes, depending on the options used. It can be issued only to a network node; it is not valid at an end node or a LEN node.

This verb must be issued to a running node.

VCB structure

```
typedef struct query_nn_topology_node
{
   AP_UINT16        opcode;               /* verb operation code           */
   unsigned char    reserv2;              /* reserved                      */
   unsigned char    format;               /* reserved                      */
   AP_UINT16        primary_rc;           /* primary return code           */
   AP_UINT32        secondary_rc;         /* secondary return code         */
   unsigned char    *buf_ptr;             /* pointer to buffer             */
   AP_UINT32        buf_size;             /* buffer size                   */
   AP_UINT32        total_buf_size;       /* total buffer size required    */
   AP_UINT16        num_entries;          /* number of entries             */
   AP_UINT16        total_num_entries;    /* total number of entries       */
   unsigned char    list_options;         /* listing options               */
   unsigned char    reserv3;              /* reserved                      */
   unsigned char    node_name[17];        /* network qualified node name   */
   unsigned char    node_type;            /* node type                     */
   AP_UINT32        frsn;                 /* flow reduction sequence number */
} QUERY_NN_TOPOLOGY_NODE;
```

If the *frsn* field is set to a non-zero value then only node entries with FRSNs equal to or greater than the one specified will be returned. If it is set to zero then all node entries are returned.

```
typedef struct nn_topology_node_summary
{
   AP_UINT16        overlay_size;         /* size of returned entry        */
   unsigned char    node_name[17];        /* network qualified node name   */
   unsigned char    node_type;            /* node type                     */
} NN_TOPOLOGY_NODE_SUMMARY;
```

```
typedef struct nn_topology_node_detail
{
   AP_UINT16        overlay_size;         /* size of returned entry        */
   unsigned char    node_name[17];        /* network qualified node name   */
   unsigned char    node_type;            /* node type                     */
   AP_UINT16        days_left;            /* days left until entry purged  */
   unsigned char    reserv1[2];           /* reserved                      */
   AP_UINT32        frsn;                 /* flow reduction sequence number */
   AP_UINT32        rsn;                  /* resource sequence number      */
   unsigned char    rar;                  /* route additional resistance   */
   unsigned char    status;               /* node status                   */
   unsigned char    function_support;     /* function support              */
   unsigned char    reserv2;              /* reserved                      */
   unsigned char    branch_aware;         /* is the node branch aware?     */
   unsigned char    reserva[19];          /* reserved                      */
} NN_TOPOLOGY_NODE_DETAIL;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_NN_TOPOLOGY_NODE

buf_ptr
A pointer to a data buffer that CS Linux will use to return the requested information.

buf_size
Size of the supplied data buffer.

num_entries
Maximum number of nodes for which data should be returned. To request data for a specific node rather than a range, specify the value 1. To return as many entries as possible, specify zero; in this

case, CS Linux will return the maximum number of entries that can be accommodated in the supplied data buffer.

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

AP_SUMMARY
Summary information only.

AP_DETAIL
Detailed information.

Combine this value using a logical OR operation with one of the following values:

AP_FIRST_IN_LIST
Start at the first entry in the list.

AP_LIST_INCLUSIVE
Start at the entry specified by the combination of the *node_name*, *node_type*, and *frsn* parameters.

AP_LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the combination of the *node_name*, *node_type*, and *frsn* parameters.

The list is ordered by *node_name*, then by *node_type* (in the order AP_NETWORK_NODE, AP_VRN), and lastly in numerical order of *frsn*. For more information about how the list is ordered and how the application can obtain specific entries from it, see “List options for QUERY_* Verbs” on page 34.

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 AP_FIRST_IN_LIST

The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

node_type
Type of the node. This value is ignored if *list_options* is set to AP_FIRST_IN_LIST. Possible values are:

AP_NETWORK_NODE
Network node.

AP_VRN
Virtual routing node.

AP_LEARN_NODE
Node type is unknown.

frsn
Flow Reduction Sequence Number (FRSN). Specify 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 the application needs to issue several verbs to obtain all the information. The application should 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 verbs as necessary to get all the database entries, with the *frsn* parameter set to 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 further QUERY_NN_TOPOLOGY_NODE verbs with the *frsn* parameter set to this new value. These verbs will return only the entries that have changed.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

nn_topology_node_summary.overlay_size
The size of the returned `nn_topology_node_summary` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `nn_topology_node_summary` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

nn_topology_node_summary.node_name
Fully qualified name of the node. This is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of 1-8 A-string characters, an EBCDIC dot (period) character, and a network name of 1-8 A-string characters.

nn_topology_node_summary.node_type
Type of the node. This is one of the following:

AP_NETWORK_NODE
Network node.

AP_END_NODE
End node.

AP_VRN
Virtual routing node.

nn_topology_node_detail.node_name
Fully qualified name of the node. This is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of 1-8 A-string characters, an EBCDIC dot (period) character, and a network name of 1-8 A-string characters.

nn_topology_node_detail.node_type
Type of the node. This is one of the following:

AP_NETWORK_NODE
Network node.

AP_END_NODE
End node.

AP_VRN
Virtual routing node.

nn_topology_node_detail.overlay_size
The size of the returned `nn_topology_node_detail` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `nn_topology_node_detail` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

nn_topology_node_detail.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 zero, indicating that this entry is never deleted.

nn_topology_node_detail.frsn
Flow Reduction Sequence Number (FRSN). Indicates the last time that this resource was updated at the local node.

nn_topology_node_detail.rsn
Resource Sequence Number. This is assigned by the network node that owns this resource.

nn_topology_node_detail.rar
The node's route additional resistance. Values are in the range `0-255`.

nn_topology_node_detail.status
Specifies the status of the node. This parameter may be set to `AP_UNCONGESTED`, to any one of the other values listed, or to two or more of the other values combined using a logical `OR`. Possible values are:

AP_UNCONGESTED
The number of ISR sessions is below the *isr_sessions_upper_threshold* value in the node's configuration.

AP_CONGESTED
The number of ISR sessions exceeds the threshold value.

AP_IRR_DEPLETED
The number of ISR sessions has reached the maximum specified for the node.

AP_ERR_DEPLETED
The number of endpoint sessions has reached the maximum specified.

AP_QUIESCING
A STOP_NODE of type `AP_QUIESCE` or `AP_QUIESCE_ISR` has been issued.

nn_topology_node_detail.function_support
Specifies which functions are supported. This may be one or more of the following, combined using a logical `OR`.

AP_BORDER_NODE
Border Node

AP_EXTENDED_BORDER_NODE
Return border node function is supported.

AP_CDS
Central Directory server

AP_GATEWAY
Gateway Node

AP_INTERCHANGE_NODE
Interchange node function is supported.

AP_ISR
Intermediate Session Routing.

AP_HPR
Node supports the base functions of High Performance Routing (HPR).

AP_RTP_TOWER
Node supports the Rapid Transport Protocol tower of HPR.

AP_CONTROL_OVER_RTP_TOWER
Node supports HPR control flows over the Rapid Transport Protocol tower.

nn_topology_node_detail.branch_aware
Specifies whether the node supports branch awareness, APPN Option Set 1120.

AP_NO
The node does not support option set 1120.

AP_YES
The node supports option set 1120.

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

ノードの追加ノード
list_options パラメーターが **リストを含む** （ **包括的** ）に設定され、指定された名前から始まるすべての項目がリストされましたが、 ノード名 パラメーターが無効でした。

ファイルの追加リスト・オプション
list_options パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: function not supported

If the verb does not execute successfully because the local node is not a network node, CS Linux returns the following parameters:

primary_rc

AP_FUNCTION_NOT_SUPPORTED
The local node is not a network node. This verb can be used only at a network node.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

QUERY_NN_TOPOLOGY_STATS

QUERY_NN_TOPOLOGY_STATS returns statistical information about the topology database. It can be used only if the CS Linux node is a network node, and is not valid if it is an end node.

This verb must be issued to a running node.

VCB 構造体

```
typedef struct query_nn_topology_stats
{
  AP_UINT16       opcode;               /* verb operation code             */
  unsigned char   reserv2;              /* reserved                        */
```

```
  unsigned char   format;                /* reserved                          */
  AP_UINT16       primary_rc;            /* primary return code               */
  AP_UINT32       secondary_rc;          /* secondary return code             */
  AP_UINT32       max_nodes;             /* max number of nodes in database   */
  AP_UINT32       cur_num_nodes;         /* current number of nodes in        */
                                         /*  database                         */
  AP_UINT32       node_in_tdus;          /* number of TDUs received           */
  AP_UINT32       node_out_tdus;         /* number of TDUs sent               */
  AP_UINT32       node_low_rsns;         /* node updates received with low    */
                                         /*  RSNs                             */
  AP_UINT32       node_equal_rsns;       /* node updates in with equal RSNs   */
  AP_UINT32       node_good_high_rsns;   /* node updates in with high RSNs    */
  AP_UINT32       node_bad_high_rsns;    /* node updates in with high and     */
                                         /*  odd RSNs                         */
  AP_UINT32       node_state_updates;    /* number of node updates sent       */
  AP_UINT32       node_errors;           /* number of node entry errors found*/
  AP_UINT32       node_timer_updates;    /* number of node records built      */
                                         /*  due to timer updates             */
  AP_UINT32       node_purges;           /* number of node records purged     */
  AP_UINT32       tg_low_rsns;           /* TG updates received with low RSNs*/
  AP_UINT32       tg_equal_rsns;         /* TG updates in with equal RSNs     */
  AP_UINT32       tg_good_high_rsns;     /* TG updates in with high RSNs      */
  AP_UINT32       tg_bad_high_rsns;      /* TG updates in with high and       */
                                         /*  odd RSNs                         */
  AP_UINT32       tg_state_updates;      /* number of TG updates sent         */
  AP_UINT32       tg_errors;             /* number of TG entry errors found   */
  AP_UINT32       tg_timer_updates;      /* number of node records built      */
                                         /*  due to timer updates             */
  AP_UINT32       tg_purges;             /* number of node records purged     */
  AP_UINT32       total_route_calcs;     /* number of routes calculated       */
                                         /*  for COS                          */
  AP_UINT32       total_route_rejs;      /* number of failed route            */
                                         /*  calculations                     */
  AP_UINT32       total_tree_cache_hits; /* total number of tree cache hits   */
  AP_UINT32       total_tree_cache_misses;   /* total number of tree cache    */
                                             /*  misses                       */
  AP_UINT32       total_tdu_wars;        /* total number TDU war detections   */
  unsigned char   reserva[16];           /* reserved                          */
} QUERY_NN_TOPOLOGY_STATS;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
追加トポロジー・トポロジーの統計

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

max_nodes
Maximum number of node records in the Topology Database. This value was specified on DEFINE_NODE. A value of 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 Topology Database Updates (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 RSN less than 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. No update to this node's topology database occurs, but this node will send a TDU with its higher RSN to the adjacent node that sent this 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. No update to this node's topology database occurs.)

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. It 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. Updates are sent via TDUs to all adjacent network nodes.

node_errors
Total number of topology node update inconsistencies detected by this node. This occurs when this node attempts to update its topology database and detects a data inconsistency. This node will create a TDU with the current RSN incremented to the next odd number and broadcast 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. 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. This 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 result when TDUs are broadcast to all adjacent network nodes. No update to this node's topology database occurs, 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 result when TDUs are broadcast to all adjacent network nodes. No update to this node's topology database occurs.)

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. Updates are sent via TDUs to all adjacent network nodes.

tg_errors
Total number of topology TG update inconsistencies detected by this node. This occurs when this node attempts to update its topology database and detects a data inconsistency. This node will create

a TDU with the current RSN incremented to the next odd number and broadcast 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. 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. This 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, since each route may require 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.

戻りパラメーター: 関数はサポートされません

ローカル・ノードがネットワーク・ノードではないために verb が正常に実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc

付加機能がサポートされていません
ローカル・ノードがネットワーク・ノードではありません。 この verb は、ネットワーク・ノードでのみ使用できます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

QUERY_NN_TOPOLOGY_TG

Each network node maintains a network topology database which holds information about all the network nodes, VRNs and network node to network node TGs in the network. QUERY_NN_TOPOLOGY_TG returns information about the TG entries in this database.

This verb can be used to obtain either summary or detailed information, about a specific TG or about multiple TGs, depending on the options used. It can be issued only to a network node; it is not valid at an end node or a LEN node.

This verb must be issued to a running node.

VCB structure

```
typedef struct query_nn_topology_tg
{
   AP_UINT16         opcode;               /* verb operation code            */
```

```
   unsigned char     reserv2;              /* reserved                        */
   unsigned char     format;               /* reserved                        */
   AP_UINT16         primary_rc;           /* primary return code             */
   AP_UINT32         secondary_rc;         /* secondary return code           */
   unsigned char     *buf_ptr;             /* pointer to buffer               */
   AP_UINT32         buf_size;             /* buffer size                     */
   AP_UINT32         total_buf_size;       /* total buffer size required      */
   AP_UINT16         num_entries;          /* number of entries               */
   AP_UINT16         total_num_entries;    /* total number of entries         */
   unsigned char     list_options;         /* listing options                 */
   unsigned char     reserv3;              /* reserved                        */
   unsigned char     owner[17];            /* node that owns the TG           */
   unsigned char     owner_type;           /* type of node that owns the TG   */
   unsigned char     dest[17];             /* TG destination node             */
   unsigned char     dest_type;            /* TG destination node type        */
   unsigned char     tg_num;               /* TG number                       */
   unsigned char     reserv1;              /* reserved                        */
   AP_UINT32         frsn;                 /* flow reduction sequence number */
} QUERY_NN_TOPOLOGY_TG;
```

```
typedef struct topology_tg_summary
{
   AP_UINT16         overlay_size;         /* size of returned entry          */
   unsigned char     owner[17];            /* node that owns the TG           */
   unsigned char     owner_type;           /* type of node that owns the TG   */
   unsigned char     dest[17];             /* TG destination node             */
   unsigned char     dest_type;            /* TG destination node type        */
   unsigned char     tg_num;               /* TG number                       */
   unsigned char     reserv3[1];           /* reserved                        */
   AP_UINT32         frsn;                 /* flow reduction sequence number */
} TOPOLOGY_TG_SUMMARY;
```

```
typedef struct topology_tg_detail
{
   AP_UINT16         overlay_size;         /* size of returned entry          */
   unsigned char     owner[17];            /* node that owns the TG           */
   unsigned char     owner_type;           /* type of node that owns the TG   */
   unsigned char     dest[17];             /* TG destination node             */
   unsigned char     dest_type;            /* TG destination node type        */
   unsigned char     tg_num;               /* TG number                       */
   unsigned char     reserv3[1];           /* reserved                        */
   AP_UINT32         frsn;                 /* flow reduction sequence number */
   AP_UINT16         days_left;            /* days left until entry purged    */
   LINK_ADDRESS      dlc_data;             /* DLC signalling data             */
   AP_UINT32         rsn;                  /* resource sequence number        */
   unsigned char     status;               /* tg status                       */
   TG_DEFINED_CHARS  tg_chars;             /* TG characteristics              */
   unsigned char     subarea_number;       /* subarea number                  */
   unsigned char     tg_type;              /* TG type                         */
   unsigned char     intersubnet_tg;       /* TG between subnets              */
   unsigned char     cp_cp_session_active; /* Are CP-CP sessions active?      */
   unsigned char     branch_tg;            /* TG branch aware?                */
   unsigned char     multilink_tg;         /* reserved                        */
   unsigned char     appended_data_format; /* format of appended data         */
   unsigned char     appended_data_len;    /* length of appended data         */
   unsigned char     reserva[9];           /* reserved                        */
} TOPOLOGY_TG_DETAIL;
```

```
typedef struct link_address
{
  unsigned char   format;                 /* type of link address            */
  unsigned char   reserve1;               /* reserved                        */
  AP_UINT16       length;                 /* length                          */
  unsigned char   address[32];            /* address                         */
} LINK_ADDRESS;
```

For details of the TG_DEFINED_CHARS structure, see “DEFINE_LS” on page 104.

If the *frsn* field is set to a non-zero value then only node entries with that FRSN or greater will be returned. If it is set to zero then all node entries are returned.

If the *list_options* parameter specifies detailed information, a TG Descriptor CV may be appended to the returned information. See the descriptions of the parameters *topology_tg_detail.appended_data_format* and *topology_tg_detail.appended_data_len* for more information.

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_NN_TOPOLOGY_TG

buf_ptr
A pointer to a data buffer that CS Linux will use to return the requested information.

buf_size
Size of the supplied data buffer.

num_entries
Maximum number of TGs for which data should be returned. To request data for a specific TG rather than a range, specify the value 1. To return as many entries as possible, specify zero; in this case, CS Linux will return the maximum number of entries that can be accommodated in the supplied data buffer.

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

AP_SUMMARY
Summary information only.

AP_DETAIL
Detailed information.

Combine this value using a logical OR operation with one of the following values:

AP_FIRST_IN_LIST
Start at the first entry in the list.

AP_LIST_INCLUSIVE
Start at the entry specified by the combination of owner, destination, TG number, and FRSN.

AP_LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the combination of owner, destination, TG number, and FRSN.

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 AP_LIST_INCLUSIVE or AP_LIST_FROM_NEXT.

The list is ordered by *owner*, *owner_type* (in the order AP_NETWORK_NODE, AP_VRN), *dest*, *dest_type* (in the order AP_NETWORK_NODE, AP_VRN), *tg_num* (numerically), and lastly *frsn* (numerically). For more information about how the list is ordered and how the application can obtain specific entries from it, see "List options for QUERY_* Verbs" on page 34.

owner
Name of the node that owns the TG. This value is ignored if *list_options* is set to AP_FIRST_IN_LIST. The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

owner_type
Type of the node that owns the TG. This parameter is ignored if *list_options* is set to AP_FIRST_IN_LIST. Possible values are:

AP_NETWORK_NODE
Network node.

AP_VRN
Virtual routing node.

AP_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 AP_FIRST_IN_LIST. The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

dest_type
Type of the destination node for the TG. This value is ignored if *list_options* is set to AP_FIRST_IN_LIST. Possible values are:

AP_NETWORK_NODE
Network node.

AP_VRN
Virtual routing node.

AP_LEARN_NODE
Node type is unknown.

tg_num
Number associated with the TG.

frsn
Flow Reduction Sequence Number (FRSN). Specify 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 the application needs to issue several verbs to obtain all the information. The application should 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_TG verbs as necessary to get all the database entries, with the *frsn* parameter set to 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 further QUERY_NN_TOPOLOGY_TG verbs with the *frsn* parameter set to this new value. These verbs will return only the entries that have changed.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

topology_tg_summary.overlay_size
The size of the returned `topology_tg_summary` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `topology_tg_summary` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

topology_tg_summary.owner
Name of the node that owns the TG. The name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of 1-8 A-string characters, an EBCDIC dot (period) character, and a network name of 1-8 A-string characters.

topology_tg_summary.owner_type
Type of the node that owns the TG. Possible values are:

AP_NETWORK_NODE
Network node.

AP_END_NODE
End node.

AP_VRN
Virtual routing node.

topology_tg_summary.dest
Name of the destination node for the TG. The name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of 1-8 A-string characters, an EBCDIC dot (period) character, and a network name of 1-8 A-string characters.

topology_tg_summary.dest_type
Type of the destination node for the TG. Possible values are:

AP_NETWORK_NODE
Network node.

AP_END_NODE
End node.

AP_VRN
Virtual routing node.

topology_tg_summary.tg_num
Number associated with the TG.

topology_tg_summary.frsn
Flow Reduction Sequence Number (FRSN), indicating the last time that this resource was updated at the local node.

topology_tg_detail.overlay_size
The size of the returned `topology_tg_detail` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `topology_tg_detail` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

topology_tg_detail.owner
Name of the node that owns the TG. The name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of 1-8 A-string characters, an EBCDIC dot (period) character, and a network name of 1-8 A-string characters.

topology_tg_detail.owner_type
Type of the node that owns the TG. Possible values are:

AP_NETWORK_NODE
Network node.

AP_END_NODE
End node.

AP_VRN
Virtual routing node.

topology_tg_detail.dest
Name of the destination node for the TG. The name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of 1-8 A-string characters, an EBCDIC dot (period) character, and a network name of 1-8 A-string characters.

topology_tg_detail.dest_type
Type of the destination node for the TG. Possible values are:

AP_NETWORK_NODE
Network node.

AP_END_NODE
End node.

AP_VRN
Virtual routing node.

topology_tg_detail.tg_num
Number associated with the TG.

topology_tg_detail.frsn
Flow Reduction Sequence Number (FRSN), indicating the last time that this resource was updated at the local node.

topology_tg_detail.days_left
Number of days before this TG entry will be deleted from the Topology Database.

topology_tg_detail.dlc_data.length
If *dest_type* or *owner_type* is AP_VRN, this field specifies the length of the DLC address in the following field. Otherwise, this field is not used.

topology_tg_detail.dlc_data.address
If *dest_type* or *owner_type* is AP_VRN, this field specifies the DLC address (in hexadecimal) of the connection to the VRN. The number of bytes in the address is given by the preceding field, length; the remaining bytes in the field are undefined. Otherwise, this field is not used.

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 124.

For Enterprise Extender (HPR/IP), see “QUERY_LS” on page 370 for details of the address format.

topology_tg_detail.rsn
Resource Sequence Number. This is assigned by the network node that owns this resource.

topology_tg_detail.status
Specifies the status of the TG. This may be one or more of the following, combined using a logical OR operation.

```
AP_NONE

AP_TG_OPERATIVE

AP_TG_QUIESCING

AP_TG_CP_CP_SESSIONS

AP_HPR

AP_RTP
```

topology_tg_detail.tg_chars
TG characteristics. For details of these parameters, see “DEFINE_LS” on page 104.

topology_tg_detail.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.

topology_tg_detail.tg_type
Type of the TG. Possible values are:

AP_APPN_OR_BOUNDARY_TG
APPN TG or boundary function based TG.

AP_INTERCHANGE_TG
Interchange TG.

AP_VIRTUAL_ROUTE_BASED_TG
Virtual route based TG.

AP_UNKNOWN
The TG type is unknown.

topology_tg_detail.intersubnet_tg
Specifies whether the TG is an intersubnetwork TG. Possible values are:

AP_YES
The TG is an intersubnetwork TG.

AP_NO
The TG is not an intersubnetwork TG.

topology_tg_detail.cp_cp_session_active
Specifies whether the owning node's contention winner CP-CP session is active. Possible values are:

AP_YES
The CP-CP session is active.

AP_NO
The CP-CP session is not active.

AP_UNKNOWN
The CP-CP session status is unknown.

topology_tg_detail.branch_tg
Specifies whether the TG is a branch TG. Possible values are:

AP_YES
The TG is a branch TG.

AP_NO
The TG is not a branch TG.

AP_UNKNOWN
The TG type is unknown.

topology_tg_detail.appended_data_format
Specifies the format of data appended to this NOF VCB structure.

If the parameter *topology_tg_detail.appended_data_len* is set to a non-zero value, indicating that appended data is included, this parameter is set to the following value:

AP_TG_DESCRIPTOR_CV
The appended data contains a TG Descriptor CV, as defined by SNA Formats.

If *topology_tg_detail.appended_data_len* is zero, indicating that no appended data is included, this parameter is reserved.

topology_tg_detail.appended_data_len
Specifies the length of the TG Descriptor CV data appended to this NOF VCB structure. If this parameter is set to zero, no appended data is included.

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_TG
The *list_options* parameter was set to AP_LIST_INCLUSIVE to list all entries starting from the supplied name, but the *tg_num* parameter was not valid.

AP_INVALID_ORIGIN_NODE
The *list_options* parameter was set to AP_LIST_INCLUSIVE to list all entries starting from the supplied name, but the *owner* parameter was not valid.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: function not supported

If the verb does not execute successfully because the local node is not a network node, CS Linux returns the following parameters:

primary_rc

AP_FUNCTION_NOT_SUPPORTED
The local node is not a network node. This verb can be used only at a network node.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

QUERY_NODE

QUERY_NODE returns information about the definition of a CS Linux node, and on its status if it is active.

VCB structure

```
typedef struct query_node
{
  AP_UINT16        opcode;                  /* verb operation code        */
  unsigned char    reserv2;                 /* reserved                   */
  unsigned char    format;                  /* reserved                   */
  AP_UINT16        primary_rc;              /* primary return code        */
  AP_UINT32        secondary_rc;            /* secondary return code      */
  CP_CREATE_PARMS  cp_create_parms;         /* create parameters          */
  AP_UINT32        up_time;                 /* time since node started    */
  AP_UINT32        mem_size;                /* reserved                   */
  AP_UINT32        mem_used;                /* reserved                   */
  AP_UINT32        mem_warning_threshold;   /* reserved                   */
  AP_UINT32        mem_critical_threshold;  /* reserved                   */
  unsigned char    nn_functions_supported;  /* NN functions supported     */
  unsigned char    functions_supported;     /* functions supported        */
  unsigned char    en_functions_supported;  /* EN functions supported     */
  unsigned char    nn_status;               /* node status                */
  AP_UINT32        nn_frsn;                 /* NN flow reduction sequence */
                                            /*  number                    */
  AP_UINT32        nn_rsn;                  /* Resource sequence number   */
  AP_UINT16        def_ls_good_xids;        /* Good XIDS for defined link */
                                            /*  stations                  */
  AP_UINT16        def_ls_bad_xids;         /* Bad XIDS for defined link  */
                                            /*  stations                  */
```

```
  AP_UINT16        dyn_ls_good_xids;           /* Good XIDS for dynamic link */
                                               /*  stations                  */
  AP_UINT16        dyn_ls_bad_xids;            /* Bad XIDS for dynamic link  */
                                               /*  stations                  */
  unsigned char    dlur_release_level;         /* Current DLUR release level */
  unsigned char    nns_dlus_served_lu_reg_supp; /* NNS supports DLUS-served  */
                                                /* LU registration?          */
  unsigned char    nns_en_reg_diff_owning_cp;   /* NNS supports option 1123? */
  unsigned char    reserva[17];                 /* reserved                  */
  unsigned char    fq_nn_server_name[17];      /* fully qualified NN server  */
                                               /*  name                      */
  AP_UINT32        current_isr_sessions;       /* number of ISR sessions     */
  unsigned char    nn_functions2;              /* further NN fns supported   */
  unsigned char    branch_ntwk_arch_version;   /* level of BrNN support      */
  unsigned char    reservb[28];                /* reserved                   */
} QUERY_NODE;
```

```
typedef struct cp_create_parms
{
  AP_UINT16       crt_parms_len;          /* length of CP_CREATE_PARMS       */
  unsigned char   description[32];        /* resource description            */
  unsigned char   reserv1[2];             /* reserved                        */
  unsigned char   ms_support;             /* reserved                        */
  unsigned char   queue_nmvts;            /* reserved                        */
  unsigned char   reserv3[12];            /* reserved                        */
  unsigned char   node_type;              /* node type                       */
  unsigned char   fqcp_name[17];          /* fully qualified CP name         */
  unsigned char   cp_alias[8];            /* CP alias                        */
  unsigned char   mode_to_cos_map_supp;   /* mode to COS mapping support     */
  unsigned char   mds_supported;          /* MDS and MS capabilities         */
  unsigned char   node_id[4];             /* node ID                         */
  AP_UINT16       max_locates;            /* maximum locates node can process*/
  AP_UINT16       dir_cache_size;         /* directory cache size            */
  AP_UINT16       max_dir_entries;        /* maximum directory entries       */
                                          /*  (0 means unlimited)            */
  AP_UINT16       locate_timeout;         /* locate timeout in seconds       */
  unsigned char   reg_with_nn;            /* register resources with NNs     */
  unsigned char   reg_with_cds;           /* register resources with CDS     */
  AP_UINT16       mds_send_alert_q_size;  /* size of MDS send alert queue    */
  AP_UINT16       cos_cache_size;         /* number of cos definitions       */
  AP_UINT16       tree_cache_size;        /* Topology Database routing tree  */
                                          /*  cache size                     */
  AP_UINT16       tree_cache_use_limit;   /* number of times a tree can be   */
                                          /*  used                           */
  AP_UINT16       max_tdm_nodes;          /* max number of nodes that can be */
                                          /*  stored in Topology Database    */
  AP_UINT16       max_tdm_tgs;            /* max number of TGs that can be   */
                                          /*  stored in Topology Database    */
  AP_UINT32       max_isr_sessions;       /* maximum ISR sessions            */
  AP_UINT32       isr_sessions_upper_threshold;  /* upper threshold for ISR  */
                                                 /*  sessions                */
  AP_UINT32       isr_sessions_lower_threshold;  /* lower threshold for ISR  */
                                                 /*  sessions                */
  AP_UINT16       isr_max_ru_size;        /* max RU size for ISR             */
  AP_UINT16       isr_rcv_pac_window;     /* ISR receive pacing window size  */
  unsigned char   store_endpt_rscvs;      /* endpoint RSCV storage           */
  unsigned char   store_isr_rscvs;        /* ISR RSCV storage                */
  unsigned char   store_dlur_rscvs;       /* DLUR RSCV storage               */
  unsigned char   dlur_support;           /* is DLUR supported?              */
  unsigned char   pu_conc_support;        /* is PU conc supported?           */
  unsigned char   nn_rar;                 /* route additional resistance     */
  unsigned char   hpr_support;            /* level of HPR support            */
  unsigned char   mobile;                 /* reserved                        */
  unsigned char   discovery_support;      /* reserved                        */
  unsigned char   discovery_group_name[8];/* reserved                        */
  unsigned char   implicit_lu_0_to_3;     /* reserved                        */
  unsigned char   default_preference;     /* reserved                        */
  unsigned char   anynet_supported;       /* reserved                        */
  AP_UINT16       max_ls_exception_events; /* max # exception entries        */
  unsigned char   reserv2[1];             /* reserved                        */
  unsigned char   max_compress_lvl;       /* Max compressson level (reserved)*/
  unsigned char   node_spec_data_len;     /* reserved                        */
  unsigned char   ptf[64];                /* program temporary fix array     */
  unsigned char   cos_table_version;      /* version of COS tables to use    */
  unsigned char   send_term_self;         /* default PLU-SLU session term    */
  unsigned char   disable_branch_awareness; /* disable BrNN awareness        */
  unsigned char   cplu_syncpt_support;    /* syncpoint support on CP LU?     */
  unsigned char   cplu_attributes;        /* attributes for CP LU            */
  unsigned char   reserved[95];           /* reserved                        */
} CP_CREATE_PARMS;
```

Supplied parameters

The application supplies the following parameter:

opcode
AP_QUERY_NODE

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

cp_create_parms.crt_parms_len
Length of create parameters structure.

cp_create_parms.description
A null-terminated text string describing the node, as specified in the definition of the node.

cp_create_parms.node_type
Type of node. Possible values are:

AP_NETWORK_NODE

AP_BRANCH_NETWORK_NODE

AP_END_NODE

AP_LEN_NODE

cp_create_parms.fqcp_name
Fully qualified name of the node. This is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of 1-8 A-string characters, an EBCDIC dot (period) character, and a network name of 1-8 A-string characters.

cp_create_parms.cp_alias
Locally used CP alias. This is an 8-byte ASCII string. All 8 bytes are significant.

cp_create_parms.mode_to_cos_map_supp
Specifies whether mode-to-COS mapping is supported by the node. 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:

AP_YES
Mode-to-COS mapping is supported.

AP_NO
Mode-to-COS mapping is not supported.

cp_create_parms.mds_supported
Specifies whether Management Services supports Multiple Domain Support and MS Capabilities. Possible values are:

AP_YES
MDS is supported.

AP_NO
MDS is not supported.

cp_create_parms.node_id
Node identifier used in XID exchange. This is a 4-byte hexadecimal string.

cp_create_parms.max_locates
Maximum number of locates that the node can process.

cp_create_parms.dir_cache_size
Network node only: Size of the directory cache.

cp_create_parms.max_dir_entries
Maximum number of directory entries. Zero indicates no limit.

cp_create_parms.locate_timeout
Specifies the time in seconds before a network search will time out. Zero indicates no timeout.

cp_create_parms.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:

AP_YES
Register resources with the NN. The end node's network node server will only forward directed locates to it.

AP_NO
Do not register resources. The network node server will forward all broadcast searches to the end node.

cp_create_parms.reg_with_cds
End node: Specifies whether the network node server is allowed to register end node resources with a Central Directory server. This field is ignored if *reg_with_nn* is set to AP_NO.

Network node: Specifies whether local or domain resources can be optionally registered with Central Directory server (AP_YES or AP_NO).

Possible values are:

AP_YES
Register resources with the CDS.

AP_NO
Do not register resources.

cp_create_parms.mds_send_alert_q_size
Size of the MDS send alert queue. If the number of queued alerts reaches this limit, CS Linux deletes the oldest alert on the queue.

cp_create_parms.cos_cache_size
Network node: Size of the COS Database weights cache (the maximum number of COS definitions required). For an end node or LEN node, this parameter is reserved.

cp_create_parms.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.

cp_create_parms.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 nodeor LEN node, this parameter is reserved.

cp_create_parms.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 nodeor LEN node, this parameter is reserved.

cp_create_parms.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 nodeor LEN node, this parameter is reserved.

cp_create_parms.max_isr_sessions
Network node: Maximum number of ISR sessions the node can participate in at once. CS Linux uses the value 100 unless a larger number has been specified. For an end nodeor LEN node, this parameter is reserved.

cp_create_parms.isr_sessions_upper_threshold **and** ***cp_create_parms.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.

cp_create_parms.isr_max_ru_size
Network node or BrNN: Maximum RU size supported for intermediate or DLUR LU-LU sessions.

End node: Maximum RU size supported for DLUR LU-LU sessions.

For a LEN node, this parameter is reserved.

cp_create_parms.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.

cp_create_parms.store_endpt_rscvs
Specifies whether RSCVs should be stored for diagnostic purposes. Possible values are:

AP_YES
Store RSCVs.

AP_NO
Do not store RSCVs.

If this field is set to AP_YES, then an RSCV will be returned on the QUERY_SESSION verb. (Setting this value to AP_YES means an RSCV will be stored for each endpoint session. This extra storage can be up to 256 bytes per session.)

cp_create_parms.store_isr_rscvs
Network node: Specifies whether RSCVs should be stored for diagnostic purposes (AP_YES or AP_NO). If this field is set to AP_YES, then an RSCV will be returned on the QUERY_ISR_SESSION verb. (Setting this value to AP_YES means an RSCV will be stored for each ISR session. This extra storage can be up to 256 bytes per session.) For an end nodeor LEN node, this parameter is reserved.

cp_create_parms.store_dlur_rscvs
Specifies whether RSCVs should be stored for diagnostic purposes (AP_YES or AP_NO). If this field is set to AP_YES, then an RSCV will be returned on the QUERY_DLUR_LU verb. (Setting this value to AP_YES means an RSCV will be stored for each PLU-SLU session. This extra storage can be up to 256 bytes per session.)

cp_create_parms.dlur_support
Specifies whether DLUR is supported. For a LEN node, this parameter is reserved. Possible values are:

AP_YES
DLUR is supported.

AP_LIMITED_DLUR_MULTI_SUBNET | AP_YES
End Node or Branch Network 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.

AP_NO
DLUR is not supported.

cp_create_parms.pu_conc_support
Specifies whether SNA gateway is supported (AP_YES or AP_NO).

cp_create_parms.nn_rar
The network node's route additional resistance.

cp_create_parms.hpr_support
Specifies the level of HPR (High Performance Routing) support provided by the node. Possible values are:

AP_NONE
No support for HPR.

AP_BASE
This node can perform automatic network routing (ANR) but cannot act as an RTP (Rapid Transport Protocol) end point for HPR sessions.

AP_RTP
This node can perform automatic network routing (ANR) and can act as an RTP (Rapid Transport Protocol) end point for HPR sessions.

AP_CONTROL_FLOWS
This node can perform all HPR functions including control flows.

cp_create_parms.max_ls_exception_events
The maximum number of LS exception events recorded by the node.

cp_create_parms.ptf
Array for configuring and controlling future program temporary fix (ptf) operation, as follows:

cp_create_parms.ptf[0]
REQDISCONT support and Mandatory Search Status support.

CS Linux normally uses REQDISCONT to deactivate limited resource host links that are no longer required by session traffic. This byte can be used to suppress use of REQDISCONT, or to modify the settings used on REQDISCONT requests sent by CS Linux. Possible values:

AP_NONE
Use the normal REQDISCONT support.

AP_SUPPRESS_REQDISCONT
Do not use REQDISCONT.

AP_OVERRIDE_REQDISCONT
Use a modified version of REQDISCONT support. If REQDISCONT is specified, it must be combined with one or both of the following values, using a logical 0R operation:

AP_REQDISCONT_TYPE
Use type "immediate" on REQDISCONT; if this value is not specified, CS Linux uses type "normal".

AP_REQDISCONT_RECONTACT
Use type "immediate recontact" on REQDISCONT; if this value is not specified, CS Linux uses type "no immediate recontact".

When CS 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, CS 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 above is combined with the following value to override the standard operation:

AP_SET_SEARCH_STATUS
Unconditionally request network searches from the NNS.

cp_create_parms.ptf[1]
ERP support. CS 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. Possible values:

AP_NONE
Use the normal processing.

AP_OVERRIDE_ERP
Process all ACTPU requests as ACTPU(cold).

cp_create_parms.ptf[2]
LU 6.2 session activation and deactivation. CS Linux normally does not include the ENQUEUE parameter on the INIT_SELF message when activating a dependent LU 6.2 session, and uses the BIS protocol prior to deactivating a limited resource LU 6.2 session. Possible values:

AP_NONE
Use the normal processing.

AP_SUPPRESS_BIS
Do not use the BIS protocol. Limited resource LU 6.2 sessions are deactivated immediately using UNBIND(cleanup).

AP_LU62_INIT_SELF_ENQUEUE
Use the old format of the INIT_SELF message, which includes the ENQUEUE parameter.

cp_create_parms.ptf[3]
APINGD support. CS Linux normally includes a partner program for the APING connectivity tester. This byte allows you to disable the APING Daemon within the node, so that requests by an APING program arriving at the node will not be processed automatically. Possible values:

AP_NONE
Include APINGD support within the node (the normal processing).

AP_EXTERNAL_APINGD
Disable APINGD within the node.

cp_create_parms.ptf[4]
LU 0-3 RU checks. This byte is used to provide workarounds for host systems that send non-standard SNA data; it should be set to AP_NONE unless you have encountered the specific problem described below.

The value AP_NONE indicates CS Linux's normal checking on LU 0-3 RUs.

If specific checks on LU 0-3 RUs have been relaxed, the following value is returned:

AP_ALLOW_BB_RQE
The SNA protocols state that BB !EB RUs on LU 0-3 PLU-SLU sessions must be RQD. Several hosts send RQE BB !EB CD - a protocol violation which CS Linux always tolerates. If this value is set, CS Linux will tolerate RQE BB !EB !CD EC RUs as well.

AP_SEND_ACTLU_POWER_ON
When an application is using an LU 0-3 LU (for example if RUI_INIT has been received for the LU) and ACTLU is received, this option indicates that CS Linux should respond with a +ve RSP ACTLU containing the power on subvector. Without this flag CS Linux sends an ACTLU RSP without this subvector and a subsequent NOTIFY message indicating the power on condition.

cp_create_parms.ptf[5]
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 CS Linux receives, CS 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 parameter allows you to disable the checking. Possible values:

AP_NONE
Always check security parameters if they are included on a received Attach, regardless of the security requirements of the invokable TP (the normal processing).

AP_LIMIT_TP_SECURITY
Do not check security parameters on a received Attach if the invokable TP does not require it.

cp_create_parms.ptf[6]
RTP options for HPR.

The value AP_NONE indicates CS Linux's normal RTP processing.

For customized RTP operation, one of the following values is returned:

AP_NO_PROGRESSIVE_ARB
If this value is set, CS Linux will advertise support for the standard and responsive mode ARB algorithms but not for the progressive mode algorithm.

cp_create_parms.ptf[7]
DLUR unbind on DACTLU. CS Linux does not normally end the PLU-SLU session when it receives a DACTLU from the host for a session using DLUR. This parameter allows you to force ending of the PLU-SLU session. Possible values:

AP_NONE
Use the normal processing.

AP_DLUR_UNBIND_ON_DACTLU
When DACTLU is received on a session using DLUR, end the PLU-SLU session.

cp_create_parms.ptf[8]
Suppress PU name on REQACTPU. CS Linux normally identifies the PU name in the REQACTPU message when activating DLUR PUs. Possible values:

AP_NONE
Use the normal processing.

AP_SUPPRESS_PU_NAME_ON_REQACTPU
Suppress PU name when activating DLUR PUs.

AP_RETRY_CNOS_ON_BIND_NEG_RSP
During APPC session activation, the CNOS session activation can fail due to transitory conditions on the partner system. Some conditions indicated by particular sense codes are always retried (with a timer). This flag indicates that CS Linux will always retry failed CNOS session activations.

cp_create_parms.ptf[9]
RUI bracket race options, limited resource override options for connection networks, and TCP/IP Information Control Vector options.

If an RUI application is using bracket protocols, and the host sends a BB (Begin Bracket) after the RUI application has already sent one, CS Linux normally rejects this with sense data of 0813 and does not pass it to the application. Possible values:

AP_NONE
Use the normal processing.

AP_LUA_PASSTHRU_BB_RACE
Pass the BB through to the RUI application. The application should send a negative response with sense data of either 0813 or 0814.

A link in CS Linux that uses a connection network is normally a limited resource. The following value overrides this default:

AP_CN_OVERRIDE_LIM_RES
Use the *implicit_limited_resource* parameter in the port associated with each connection network link to determine whether it is a limited resource.

CS 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. For TN3270, the TN3270 server normally performs a Domain Name Server (DNS) lookup to determine the client IP name.

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.

The following flags allow you to override this behavior.

AP_NO_TCPIP_VECTOR
Do not include the TCP/IP Information Control Vector (0x64) in NOTIFY requests to the host for either TN3270 or LUA.

Use this value if the host is running an older version of VTAM that does not support this control vector.

AP_NO_TCPIP_NAME
Do not perform the DNS lookup, and 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. Use this value 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).

cp_create_parms.ptf[10]
Suppress Logical Unit of Work Identifiers (LUWIDs) in FMH-5 Attach messages. CS Linux normally includes the LUWID in the FMH-5 Attach message that it sends to start an APPC conversation.

The following flag allows you to override this behavior.

AP_DONT_SEND_LUWIDS
Do not include the LUWID in the FMH-5 Attach; specify the field length for this field as zero.

cp_create_parms.ptf[11]
LU management options. Possible values:

AP_NONE
Use the normal processing.

AP_DLUR_USE_REX_PACING
When the BIND from an upstream LU requests adaptive pacing with an unlimited pacing window,CS Linux normally indicates this by specifying a window size of 0 (zero). If the downstream LU does not support adaptive pacing, it may incorrectly interpret this zero value as "no pacing", so CS Linux must specify a non-zero pacing window size instead. This option indicates that CS Linux uses the REX stage pacing value from the ACTLU as the pacing window size specified to the downstream LU.

AP_CLI_OVERWRITE_SYS_NAME
This option indicates that CS Linux maintains the association between an APPC application running on a client and the pooled LU that it uses, so that subsequent conversations started by the partner application can be routed to the correct client. When a client application accesses an LU in a pool, CS Linux changes the *sys_name* parameter on the LU to the hostname of the client computer where the application is running. For more information about managing clients, refer to the *IBM Communications Server for Data Center Deployment on Linux Administration Guide*.

AP_OVERWRITE_INTERNAL_PU_PARMS
Normally ,once a DLUR PU is defined, there is no way to modify configuration parameters on that PU without first deleting it (and any associated LUs). This flag permits CS Linux to accept a fresh definition of a DLUR PU using `snaadmin` providing also that the node is inactive. All non-defaulted parameters must be defined (this is not equivalent to a `snaadmin -c` command).

cp_create_parms.cos_table_version
Specifies the version of the COS tables used by the node. Possible values:

AP_VERSION_0_COS_TABLES
Use the COS tables originally defined in the APPN Architecture Reference.

AP_VERSION_1_COS_TABLES
Use the COS tables originally defined for HPR over ATM.

cp_create_parms.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. Possible values:

AP_YES
Send a TERM_SELF on receipt of a CLOSE_PLU_SLU_SEC_RQ.

AP_NO
Send an UNBIND on receipt of a CLOSE_PLU_SLU_SEC_RQ.

cp_create_parms.disable_branch_awareness
This parameter applies only if *node_type* is AP_NETWORK_NODE; it is reserved for other node types.

Specifies whether the local node supports branch awareness, APPN Option Set 1120. Possible values:

AP_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.

AP_NO
The local node supports branch awareness.

cp_create_parms.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 AP_YES only if you have a Sync Point Manager (SPM) and Conversation Protected Resource Manager (C-PRM) in addition to the standard CS Linux product. Possible values are:

AP_YES
Syncpoint is supported.

AP_NO
Syncpoint is not supported.

cp_create_parms.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:

AP_NONE
No additional information identified.

AP_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. CS 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). The option should not be set unless there are problems with the remote system's implementation of password substitution.

up_time
Time (in hundredths of a second) since the node was started (or restarted). A value of zero indicates that the node is not running.

nn_functions_supported
Network node only: Specifies the network node functions supported. This may be one or more of the following, combined using a logical OR.

AP_RCV_REG_CHAR
Node supports receiving registered characteristics.

AP_GATEWAY
Node is a gateway node.

AP_CDS
Node supports Central Directory server function.

AP_TREE_CACHING
Node supports route tree cache.

AP_TREE_UPDATES
Node supports incremental tree updates. If this is supported, tree caching must also be supported.

AP_ISR
Node supports Intermediate Session Routing.

functions_supported
Specifies the functions supported. This may be one or more of the following, combined using a logical OR.

AP_NEGOTIABLE_LS

AP_SEGMENT_REASSEMBLY

AP_BIND_REASSEMBLY

AP_PARALLEL_TGS

AP_CALL_IN

AP_ADAPTIVE_PACING

AP_TOPOLOGY_AWARENESS

en_functions_supported
End node only: Specifies the end node functions supported. This may be one or more of the following, combined using a logical OR.

AP_SEGMENT_GENERATION
Node supports segment generation.

AP_MODE_TO_COS_MAP
Node supports mode name to COS name mapping.

AP_LOCATE_CDINIT
Node supports generation of locates and cross-domain initiate GDS variables for locating remote LUs.

AP_REG_WITH_NN
Node will register its LUs with the adjacent serving network node.

AP_REG_CHARS_WITH_NN
Node supports send register characteristics. If this function is supported, send registered names must also be supported.

nn_status
Network node only: Specifies the status of the node. This parameter may be set to AP_UNCONGESTED, to any one of the other values listed, or to two or more of the other values combined using a logical OR. Possible values are:

AP_UNCONGESTED
The number of ISR sessions is below the *isr_sessions_upper_threshold* value in the node's configuration.

AP_CONGESTED
The number of ISR sessions exceeds the threshold value.

AP_IRR_DEPLETED
The number of ISR sessions has reached the maximum specified for the node.

AP_ERR_DEPLETED
The number of endpoint sessions has reached the maximum specified.

AP_QUIESCING
A STOP_NODE of type AP_QUIESCE or AP_QUIESCE_ISR has been issued.

nn_frsn
Network node only: The network node's current Flow Reduction Sequence Number (FRSN).

nn_rsn
Network node only: Resource sequence number.

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 is set to the value 1 (the only release level of DLUR currently defined); future versions may incorporate later release levels of the DLUR architecture, and so may return different values.

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:

AP_YES
The network node server supports registration of DLUS-served LUs.

AP_NO
The network node server does not support registration of DLUS-served LUs.

AP_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.

AP_YES
The network node server supports option set 1123.

AP_NO
The network node server does not support option set 1123.

AP_UNKNOWN
The node does not have a network node server.

fq_nn_server_name
End node only. Name of the network node server for the node.

current_isr_sessions
This parameter applies only if the local node is a Network Nodeor a Branch Network Node; it is reserved otherwise.

Number of ISR sessions routed through this node.

nn_functions_2
This parameter applies only if the local node is a Network Node; it is reserved otherwise.

Specifies whether the node supports branch awareness, APPN Option Set 1120.

AP_NONE
The network node server does not support option set 1120.

AP_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.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

QUERY_NODE_ALL

QUERY_NODE_ALL returns information about nodes on the CS Linux LAN. This verb returns only each node's name and configuration file role, and does not provide detailed information about the node's configuration. The application can use QUERY_NODE for a particular node name to obtain detailed information about that node.

This verb must be issued with a null target handle.

VCB 構造体

```
typedef struct query_node_all
{
   AP_UINT16       opcode;                /* verb operation code            */
   unsigned char   reserv2;               /* reserved                       */
   unsigned char   format;                /* reserved                       */
   AP_UINT16       primary_rc;            /* primary return code            */
   AP_UINT32       secondary_rc;          /* secondary return code          */
   unsigned char   *buf_ptr;              /* pointer to buffer              */
   AP_UINT32       buf_size;              /* buffer size                    */
   AP_UINT32       total_buf_size;        /* total buffer size required     */
   AP_UINT16       num_entries;           /* number of entries              */
   AP_UINT16       total_num_entries;     /* total number of entries        */
   unsigned char   list_options;          /* listing options                */
   unsigned char   reserv3;               /* reserved                       */
   unsigned char   node_name[128];        /* node name                      */
} QUERY_NODE_ALL;
```

```
typedef struct node_summary
{
   AP_UINT16       overlay_size;          /* size of returned entry         */
   unsigned char   node_name[128];        /* node name                      */
   unsigned char   config_role;           /* server's config file role      */
   unsigned char   reserv3[12];           /* reserved                       */
} NODE_SUMMARY;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_NODE_ALL

buf_ptr
A pointer to a data buffer that CS Linux will use to return the requested information.

buf_size
Size of the supplied data buffer.

num_entries
Maximum number of nodes for which data should be returned. To request data for a specific node rather than a range, specify the value 1. To return as many entries as possible, specify zero; in this case, CS Linux will return the maximum number of entries that can be accommodated in the supplied data buffer.

list_options
The position in the list from which CS Linux should begin to return data. Possible values are:

AP_FIRST_IN_LIST
Start at the first entry in the list of nodes.

AP_LIST_INCLUSIVE
Start at the entry specified by the *node_name* parameter.

AP_LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *node_name* parameter.

The list is not ordered by node name. However, the order remains the same for subsequent QUERY_NODE_ALL verbs, so the application can obtain a complete list in several sections by using multiple verbs in the normal way. For more information about how the application can obtain specific entries from the list, see “List options for QUERY_* Verbs” on page 34.

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 AP_FIRST_IN_LIST.

This is an ASCII string of 1-128 characters, padded on the right with spaces if the name is shorter than 128 characters.

If the computer name includes a . (period) character, CS Linux assumes that it is a fully-qualified name; otherwise it performs a DNS lookup to determine the computer name.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

buf_size
提供されたバッファーに戻された情報の長さ。

total_buf_size
要求されたすべてのリスト情報を戻すために必要となるバッファーのサイズを示す戻り値。 *buf_size* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

num_entries
データ・バッファーに戻されたエントリーの数。

total_num_entries
使用可能な項目の合計数。 *num_entries* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

データ・バッファー内の各項目は、以下のパラメーターで構成されます。

ノード・サマリー . オーバー・サイズ
戻された **ノードの要約** 構造体のサイズ。すなわち、データ・バッファー内の次のエントリーの先頭までのオフセット。

アプリケーションが戻されたバッファーを調べて、各 **ノードの要約** 構造体を順番に検出する必要がある場合は、この値を使用して次のデータ構造の正しいオフセットに移動しなければなりません。また、 C `sizeof()` 演算子を使用してはなりません。 これは、返されるオーバーレイのサイズが、 CS Linux の将来のリリースで増加する可能性があるためです。戻されたオーバーレイ・サイズを使用すると、アプリケーションが今後のリリースで引き続き動作することが保証されます。

ノード・サマリー . ノード名
CS Linux ノードの名前。

node_summary.config_role
ノードが実行されているサーバーの構成ファイルの役割。 構成ファイルの役割について詳しくは、「*IBM Communications Server for Linux* 管理ガイド上のデータ・センター・デプロイメント」を参照してください。 可能な値は次のとおりです

AP_ROLE_CONTROLLERR
サーバーは制御構成ファイルを保持します。

追加のバックアップ （ _C ）
サーバーはバックアップ構成ファイルを保持します。

追加なし _ROLE_NONE
サーバーは、構成ファイルのコピーを共有しません。

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_NODE_NAME
The *list_options* parameter was set to AP_LIST_INCLUSIVE to list all entries starting from the supplied name, but the *node_name* parameter was not valid.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

QUERY_NODE_LIMITS

QUERY_NODE_LIMITS returns information about the functions that your CS 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 CS Linux features that you can use
- Node resource usage, which specifies the current and peak usage of CS Linux resources.

You can use the information returned by this verb to check whether your usage of CS Linux resources is within the limits permitted by your license. For more information about licensing requirements, see *IBM Communications Server for Data Center Deployment on Linux Quick Beginnings*.

The information returned by this verb is also written to the usage log file at intervals. For more information about this file, see *IBM Communications Server for Data Center Deployment on Linux Diagnostics Guide*.

VCB structure

```
typedef struct query_node_limits
{
  AP_UINT16             opcode;        /* verb operation code               */
  unsigned char         reserv2;       /* reserved                          */
  unsigned char         format;        /* reserved                          */
  AP_UINT16             primary_rc;    /* primary return code               */
```

```
  AP_UINT32             secondary_rc;  /* secondary return code              */
  NODE_RESOURCE_LIMITS  max_limits;    /* reserved                           */
  NODE_RESOURCE_LIMITS  curr_usage;    /* current usage of LUs/sessions/users*/
  NODE_OPTIONS          node_options;  /* permitted functions                */
  unsigned char         reserv4[4];    /* reserved                           */
  NODE_RESOURCE_LIMITS  max_usage;     /* highest usage counts               */
} QUERY_NODE_LIMITS;
```

```
typedef struct node_resource_limits
{
  AP_INT32     lu62_tps;                    /* APPC/CPI-C applications      */
  AP_INT32     lua_tps;                     /* LUA applications             */
  AP_INT32     fmapi_tps;                   /* reserved                     */
  AP_INT32     link_stations;               /* Active link stations         */
  AP_INT32     tn3270_connections;          /* TN3270 server connections    */
  AP_INT32     tn_redirector_connections;   /* TN redirector connections    */
  AP_INT32     v4_sna_channels;             /* reserved                     */
  AP_INT32     v4_gsna_channels;            /* reserved                     */
  AP_INT32     data_sessions;               /* Active PLU-SLU sessions      */
  AP_INT32     reserv1[11];                 /* Reserved                     */
} NODE_RESOURCE_LIMITS;
```

```
typedef struct node_options
{
   unsigned char   network_node;         /* is Network Node supported?       */
   unsigned char   end_node;             /* is End Node supported?           */
   unsigned char   len_node;             /* is LEN Node supported?           */
   unsigned char   dlur_support;         /* is DLUR supported?               */
   unsigned char   pu_conc_support;      /* is PU Conc supported?            */
   unsigned char   tn_server_support;    /* is TN Server supported?          */
   unsigned char   hpr_support;          /* level of HPR support             */
   unsigned char   back_level_client;    /* reserved                         */
   unsigned char   reserv2;              /* reserved                         */
   unsigned char   ssl_support;          /* is SSL supported?                */
   unsigned char   branch_network_node;  /* is BrNN supported?               */
   unsigned char   reserv1[21];          /* reserved                         */
} NODE_OPTIONS;
```

Supplied parameters

The application supplies the following parameter:

opcode
AP_QUERY_NODE_LIMITS

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

curr_usage.lu62_tps
このノードで現在アクティブになっている APPC アプリケーションおよび CPI-C アプリケーションの数。

curr_usage.lua_tps
このノードで現在アクティブになっている LUA アプリケーションの数。

curr_usage.link_station
このノードで現在アクティブになっているリンク・ステーションの数。

curr_usage.tn3270_connections
このノードで現在アクティブになっている TN3270 クライアントからの接続の数。

curr_usage.tn_redirector_connections
このノードで現在アクティブになっている TN リダイレクター・クライアントからの接続の数。

curr_usage.data_sessions
このノードで現在アクティブになっている PLU-SLU セッションの数。

全二重 APPC 会話を使用している場合は、各全二重会話には 2 つのセッションが必要であることに注意

最大使用率 . *lu62_tps*
Linux コンピューターの再始動以降、このノードでアクティブになっている APPC アプリケーションおよび CPI-C アプリケーションの最大数。

max_usage.lua_tps
Linux コンピューターが再始動されてから、このノードでアクティブになっている LUA アプリケーションの最大数。

***max_usage.link_*停車場**
Linux コンピューターの再始動以降、このノード上でいつでもアクティブになっているリンク・ステーションの最大数。

最大使用率 . 接続数
Linux コンピューターの再始動以降、このノードでアクティブになっている TN3270 クライアントからの接続の最大数。

max_usage.tn_redirector_connections
Linux コンピューターの再始動以降、このノードでアクティブになっている TN リダイレクター・クライアントからの接続の最大数。

max_usage.data_sessions
Linux コンピューターの再始動以降、このノードでアクティブになっていた PLU-SLU セッションの最大数。

全二重 APPC 会話を使用している場合は、各全二重会話には 2 つのセッションが必要であることに注意

node_options.network_node
ライセンスでこのノードをネットワーク・ノードとして定義できるかどうかを指定します。 可能な値は次のとおりです

類人猿
ネットワーク・ノードがサポートされる

アプ・ノー
ネットワーク・ノードがサポートされていない

ノード・オプション . end_node
ライセンスでこのノードをエンド・ノードとして定義できるかどうかを指定します。 可能な値は次のとおりです

類人猿
終了ノードがサポートされます

アプ・ノー
終了ノードはサポートされません。

ノード・オプション . len_node
ご使用のライセンスでこのノードを LEN ノードとして定義できるかどうかを指定します。 可能な値は次のとおりです

類人猿
LEN ノードがサポートされている

アプ・ノー
LEN ノードはサポートされません。

node_options.dlur_support
このパラメーターは予約済みです。

このノードで従属 LU リクエスター (DLUR) を使用することがライセンスで許可されているかどうかを指定します。 可能な値は次のとおりです

類人猿
DLUR がサポートされている

アプ・ノー
DLUR はサポートされません。

node_options.pu_conc_support
ライセンスで、このノードで SNA ゲートウェイを使用できるかどうかを指定します。 可能な値は次のとおりです

類人猿
SNA ゲートウェイがサポートされる。

アプ・ノー
SNA ゲートウェイはサポートされません。

node_options.tn_server_support
ライセンスで、このノードで TN サーバーを使用できるかどうかを指定します。 可能な値は次のとおりです

類人猿
TN サーバーはサポートされます。

アプ・ノー
TN サーバーはサポートされません。

node_options.hpr_support
ライセンスでこのノードで HPR (ハイパフォーマンス・ルーティング) を使用できるかどうかを指定します。 可能な値は次のとおりです

類人猿
HPR がサポートされている

アプ・ノー
HPR はサポートされない。

node_options.ssl_support
Secure Sockets Layer ソフトウェアがノードにインストールされているかどうかを指定します (TN サーバーと共に使用します) 。 可能な値は次のとおりです

類人猿
SSL ソフトウェアがインストールされます。

アプ・ノー
SSL ソフトウェアがインストールされていません。

node_options.branch_network_node
ライセンスでこのノードをブランチ・ネットワーク・ノードとして定義できるかどうかを指定します。 可能な値は次のとおりです

類人猿
分岐ネットワーク・ノードがサポートされます

アプ・ノー
分岐ネットワーク・ノードはサポートされない。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

照会パートナー L_LU

QUERY_PARTNER_LU は、ローカル LU が現在使用している、または使用されているパートナー LU に関する情報を戻します。 この verb は、パートナー LU の定義についてではなく、パートナー LU の使用に関する情報を戻します。 QUERY_PARTNER_LU_DEFINITION を使用して、パートナー LU の定義を取得します。

この verb は、使用されるオプションに応じて、特定の LU または複数の LU に関する要約情報または詳細情報を取得するために使用できます。

この verb は実行中のノードに対して発行する必要があります

VCB 構造体

```
typedef struct query_partner_lu
{
   AP_UINT16       opcode;               /* verb operation code             */
   unsigned char   reserv2;              /* reserved                        */
   unsigned char   format;               /* reserved                        */
   AP_UINT16       primary_rc;           /* primary return code             */
   AP_UINT32       secondary_rc;         /* secondary return code           */
   unsigned char   *buf_ptr;             /* pointer to buffer               */
   AP_UINT32       buf_size;             /* buffer size                     */
   AP_UINT32       total_buf_size;       /* total buffer size required      */
   AP_UINT16       num_entries;          /* number of entries               */
   AP_UINT16       total_num_entries;    /* total number of entries         */
   unsigned char   list_options;         /* listing options                 */
   unsigned char   reserv3;              /* reserved                        */
   unsigned char   lu_name[8];           /* LU name                         */
   unsigned char   lu_alias[8];          /* LU alias                        */
   unsigned char   plu_alias[8];         /* partner LU alias                */
   unsigned char   fqplu_name[17];       /* fully qualified partner LU name */
   unsigned char   active_sessions;      /* active sessions only filter     */
} QUERY_PARTNER_LU;
```

```
typedef struct plu_summary
{
   AP_UINT16       overlay_size;         /* size of returned entry          */
   unsigned char   plu_alias[8];         /* partner LU alias                */
   unsigned char   fqplu_name[17];       /* fully qualified partner LU name */
   unsigned char   reserv1;              /* reserved                        */
   unsigned char   description[32];      /* resource description            */
   unsigned char   reserv2[16];          /* reserved                        */
   AP_UINT16       act_sess_count;       /* currently active sessions count */
   unsigned char   partner_cp_name[17];  /* partner LU CP name              */
   unsigned char   partner_lu_located;   /* CP name resolved?               */
} PLU_SUMMARY;
```

```
typedef struct plu_detail
{
   AP_UINT16       overlay_size;         /* size of returned entry          */
   unsigned char   plu_alias[8];         /* partner LU alias                */
   unsigned char   fqplu_name[17];       /* fully qualified partner LU name */
   unsigned char   reserv1;              /* reserved                        */
   unsigned char   description[32];      /* resource description            */
   unsigned char   reserv2[16];          /* reserved                        */
   AP_UINT16       act_sess_count;       /* currently active sessions count */
   unsigned char   partner_cp_name[17];  /* partner LU CP name              */
   unsigned char   partner_lu_located;   /* CP name resolved?               */
   unsigned char   plu_un_name[8];       /* partner LU uninterpreted name   */
   unsigned char   parallel_sess_supp;   /* parallel sessions supported?    */
   unsigned char   conv_security;        /* conversation security           */
   AP_UINT16       max_mc_ll_send_size;  /* maximum send LL size for mapped */
                                         /*  conversations                  */
   unsigned char   implicit;             /* implicit or explicit entry      */
   unsigned char   security_details;     /* session security details        */
   unsigned char   duplex_support;       /* full-duplex support             */
   unsigned char   preference;           /* reserved                        */
  unsigned char    reserva[16];          /* reserved                        */
} PLU_DETAIL;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_PARTNER_LU

buf_ptr
A pointer to a data buffer that CS Linux will use to return the requested information.

buf_size
Size of the supplied data buffer.

num_entries
Maximum number of LUs for which data should be returned. To request data for a specific LU rather than a range, specify the value 1. To return as many entries as possible, specify zero; in this case, CS Linux will return the maximum number of entries that can be accommodated in the supplied data buffer.

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

AP_SUMMARY
Summary information only.

AP_DETAIL
Detailed information.

Combine this value using a logical OR operation with one of the following values:

AP_FIRST_IN_LIST
Start at the first entry in the list of partner LUs associated with the specified local LU.

AP_LIST_INCLUSIVE
Start at the entry specified by the combination of local and partner LU names.

AP_LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the combination of local and partner LU names.

AP_LIST_BY_ALIAS
The list is returned in order of LU alias rather than LU name. This option is only valid if AP_FIRST_IN_LIST is also specified. (For AP_LIST_FROM_NEXT or AP_LIST_INCLUSIVE, the list is in order of LU alias or LU name, depending on which was specified as the index into the list.)

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 partner LUs if the *list_options* parameter is set to AP_LIST_INCLUSIVE or AP_LIST_FROM_NEXT.

The list is ordered by *fqplu_name*. For more information about how the list is ordered and how the application can obtain specific entries from it, see "List options for QUERY_* Verbs" on page 34.

lu_name
LU name of the local LU. This is an 8-byte type-A EBCDIC string, padded on the right with spaces if the name is shorter than 8 characters. To indicate that the LU is identified by its LU alias instead of its LU name, set this parameter to 8 binary zeros and specify the LU alias in the following parameter.

lu_alias
LU alias of the local LU. This parameter is used only if the *lu_name* field is set to 8 binary zeros, and is ignored otherwise. The alias is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes. To indicate the LU associated with the local CP (the default LU), set both *lu_name* and *lu_alias* to binary zeros.

plu_alias
Partner LU alias. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes. If *list_options* is set to AP_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, set this parameter to 8 binary zeros and specify the LU name in the following parameter.

fqplu_name
17-byte fully qualified network name for the partner LU. If *list_options* is set to AP_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* field is set to 8 binary zeros, and is ignored otherwise.

The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

active_sessions
Specifies whether to return information only on partner LUs for which sessions are active, or on all partner LUs. Possible values are:

AP_YES
Return information only on partner LUs for which sessions are currently active.

AP_NO
Return information about all partner LUs for which sessions are active or have been active.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

plu_summary.overlay_size
The size of the returned `plu_summary` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `plu_summary` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

plu_summary.plu_alias
Partner LU alias. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes.

plu_summary.fqplu_name
17-byte fully qualified network name for the partner LU. The name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of 1-8 A-string characters, an EBCDIC dot (period) character, and a network name of 1-8 A-string characters.

plu_summary.description
A null-terminated text string describing the partner LU, as specified in the definition of the partner LU.

plu_summary.act_sess_count
Total number of active sessions between the local LU and the partner LU.

plu_summary.partner_cp_name
17-byte fully qualified network name for the CP associated with the partner LU. This parameter is not used if *partner_lu_located* below is set to AP_NO.

The name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

plu_summary.partner_lu_located
Specifies whether the local node has located the CP where the partner LU is located. Possible values are:

AP_YES
The partner LU has been located. The *partner_cp_name* parameter contains the CP name of the partner LU.

AP_NO
The partner LU has not yet been located. The *partner_cp_name* parameter should not be checked.

plu_detail.overlay_size
The size of the returned `plu_detail` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `plu_detail` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

plu_detail.plu_alias
Partner LU alias. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes.

plu_detail.fqplu_name
17-byte fully qualified network name for the partner LU. The name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of 1-8 A-string characters, an EBCDIC dot (period) character, and a network name of 1-8 A-string characters.

plu_detail.description
A null-terminated text string describing the partner LU, as specified in the definition of the partner LU.

plu_detail.act_sess_count
Total number of active sessions between the local LU and the partner LU.

plu_detail.partner_cp_name
17-byte fully qualified network name for the CP associated with the partner LU. This parameter is not used if *partner_lu_located* below is set to AP_NO.

The name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

plu_detail.partner_lu_located
Specifies whether the local node has located the CP where the partner LU is located. Possible values are:

AP_YES
The partner LU has been located. The *partner_cp_name* parameter contains the CP name of the partner LU.

AP_NO
The partner LU has not yet been located. The *partner_cp_name* parameter should not be checked.

plu_detail.plu_un_name
Uninterpreted name of the partner LU. This is an 8-byte type-A EBCDIC character string.

plu_detail.parallel_sess_supp
Specifies whether parallel sessions are supported. Possible values are:

AP_YES
Parallel sessions are supported.

AP_NO
Parallel sessions are not supported.

plu_detail.conv_security
Specifies whether conversation security information can be sent to this partner LU. Possible values are:

AP_YES
Conversation security information supplied by a local TP is sent to the partner LU.

AP_NO
Conversation security information supplied by a local TP is not sent to the partner LU.

AP_UNKNOWN
No sessions are active with the partner LU.

plu_detail.max_mc_ll_send_size
Maximum logical record size, in bytes, that can be sent to the partner LU. This may be in the range 1-32,767, or zero to indicate no limit (in which case the maximum is 32,767). Data records that are larger than this are broken down into several LL records before being sent to the partner LU.

plu_detail.implicit
Specifies whether the entry was created by an implicit or explicit definition. Possible values are:

AP_YES
The entry is an implicit entry.

AP_NO
The entry is an explicit entry.

plu_detail.security_details
Returns the conversation security support as negotiated on the BIND. Possible values are:

AP_CONVERSATION_LEVEL_SECURITY
Conversation security information will be accepted on requests to or from the partner LU to allocate a conversation. The specific types of conversation security support are described by the following values.

AP_ALREADY_VERIFIED
Both the 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.

AP_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.

AP_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 AP_PGM_STRONG will fail.

AP_UNKNOWN
No sessions are active with the partner LU.

plu_detail.duplex_support
Returns the conversation duplex support as negotiated on the BIND. Possible values are:

AP_HALF_DUPLEX
Only half-duplex conversations are supported.

AP_FULL_DUPLEX
Both full-duplex and half-duplex sessions are supported. Expedited data is also supported.

AP_UNKNOWN
The conversation duplex support is not known because no sessions are active with the partner LU.

preference
This parameter is reserved.

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_LU_ALIAS
The *list_options* parameter was set to AP_LIST_INCLUSIVE to list all entries starting from the supplied name, but the *lu_alias* parameter was not valid.

AP_INVALID_LU_NAME
The *list_options* parameter was set to AP_LIST_INCLUSIVE to list all entries starting from the supplied name, but the *lu_name* parameter was not valid.

AP_INVALID_PLU_NAME
The *list_options* parameter was set to AP_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.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

QUERY_PARTNER_LU_DEFINITION

QUERY_PARTNER_LU_DEFINITION returns information about partner LUs for a local LU. This verb returns information about the definition of the LUs, not about their current usage; use QUERY_PARTNER_LU to obtain the usage information.

This verb can be used to obtain either summary or detailed information, about a specific LU or about multiple LUs, depending on the options used.

VCB 構造体

```
typedef struct query_partner_lu_definition
{
   AP_UINT16       opcode;              /* verb operation code              */
   unsigned char   reserv2;             /* reserved                         */
   unsigned char   format;              /* reserved                         */
   AP_UINT16       primary_rc;          /* primary return code              */
   AP_UINT32       secondary_rc;        /* secondary return code            */
   unsigned char   *buf_ptr;            /* pointer to buffer                */
   AP_UINT32       buf_size;            /* buffer size                      */
   AP_UINT32       total_buf_size;      /* total buffer size required       */
   AP_UINT16       num_entries;         /* number of entries                */
   AP_UINT16       total_num_entries;   /* total number of entries          */
   unsigned char   list_options;        /* listing options                  */
   unsigned char   reserv3;             /* reserved                         */
   unsigned char   plu_alias[8];        /* partner LU alias                 */
```

```
   unsigned char   fqplu_name[17];       /* fully qualified partner LU name  */
} QUERY_PARTNER_LU_DEFINITION;
```

```
typedef struct partner_lu_def_summary
{
   AP_UINT16       overlay_size;         /* size of returned entry           */
   unsigned char   plu_alias[8];         /* partner LU alias                 */
   unsigned char   fqplu_name[17];       /* fully qualified partner LU name  */
   unsigned char   description[32];      /* resource description             */
   unsigned char   reserv1[16];          /* reserved                         */
} PARTNER_LU_DEF_SUMMARY;
```

```
typedef struct partner_lu_def_detail
{
   AP_UINT16       overlay_size;         /* size of returned entry           */
   unsigned char   plu_alias[8];         /* partner LU alias                 */
   unsigned char   fqplu_name[17];       /* fully qualified partner LU name  */
   unsigned char   reserv1;              /* reserved                         */
   PLU_CHARS       plu_chars;            /* partner LU characteristics       */
} PARTNER_LU_DEF_DETAIL;
```

```
typedef struct plu_chars
{
   unsigned char   fqplu_name[17];       /* fully qualified partner LU name  */
   unsigned char   plu_alias[8];         /* partner LU alias                 */
   unsigned char   description[32];      /* resource description             */
   unsigned char   reserv2[16];          /* reserved                         */
   unsigned char   plu_un_name[8];       /* partner LU uninterpreted name    */
   unsigned char   preference;           /* reserved                         */
   AP_UINT16       max_mc_ll_send_size;  /* maximum MC send LL size          */
   unsigned char   conv_security_ver;    /* already-verified security        */
                                         /*  supported?                      */
   unsigned char   parallel_sess_supp;   /* parallel sessions supported?     */
   unsigned char   reserv3[8];           /* reserved                         */
} PLU_CHARS;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
パートナー・ LU_LU_定義の追加

buf_ptr
CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

num_entries
データが戻される LU の最大数。 特定の範囲ではなく、特定の LU のデータを要求するには、値 1 を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、 CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータの戻りを開始する必要のあるリスト内の位置、および各項目に必要な情報のレベル。 以下の値のいずれかを使用して、情報のレベルを指定します。

要約の要約
サマリー情報のみ。

追加の詳細
詳細情報。

以下のいずれかの値を指定して、論理 **それとも** 操作を使用してこの値を結合します。

リストの最初のリスト （ _R ）
リストの最初の項目から開始します。

リストを含む （ 包括的 ）
plu_alias パラメーターまたは *fqplu_name* パラメーターで指定したエントリーから開始します。

次への AP_LIST_FROM_NEXT
plu_alias パラメーターまたは *fqplu_name* パラメーターによって指定されたエントリーの直後のエントリーから開始します。

リストの最初のリスト （ _R ）が指定されている場合は、論理 **それとも** 操作を使用して、以下のオプションを組み込むこともできます。

エイプ・リストの別名
このリストは、 LU 名ではなく LU 別名の順序で戻されます。 このオプションは、 **リストの最初のリスト** （ _R ）も指定されている場合にのみ有効です。 (**次への** AP_LIST_FROM_NEXT または **リストを含む** （ **包括的**)の場合、リストは、リスト内の索引として指定された LU 別名または LU 名の順になっています。)

アプリケーションがリストから特定の項目を取得する方法について詳しくは、 34 ページの『List options for QUERY_* Verbs』を参照してください。

plu_alias
パートナー LU の別名。 この名前は 8 バイトの ASCII ストリングで、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。 *list_options* が **リストの最初のリスト** （ _R)に設定されている場合、このパラメーターは無視されます。それ以外の場合は、パートナー LU の LU 別名または完全修飾 LU 名のいずれかを指定する必要があります。 パートナー LU がその別名の代わりに完全修飾名で定義されていることを示すには、このパラメーターを 8 個の 2 進ゼロに設定し、以下のパラメーターに名前を指定します。

fqplu_name
情報が必要なパートナー LU の完全修飾名、または LU のリストへの索引として使用される名前。 *list_options* が **リストの最初のリスト** （ _R)に設定されている場合、このパラメーターは無視されます。それ以外の場合は、パートナー LU の LU 別名または完全修飾 LU 名のいずれかを指定する必要があります。 このパラメーターは、 *plu_alias* パラメーターがゼロに設定されている場合のみ使用され、それ以外の場合は無視されます

この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

partner_lu_def_summary.overlay_size
The size of the returned `partner_lu_def_summary` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `partner_lu_def_summary` structure in turn, it must use this value to move to the correct offset for

the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

partner_lu_def_summary.plu_alias
Partner LU alias. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes.

partner_lu_def_summary.fqplu_name
Fully qualified network name for the partner LU. This name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of 1-8 A-string characters, an EBCDIC dot (period) character, and a network name of 1-8 A-string characters.

partner_lu_def_summary.description
A null-terminated text string describing the partner LU, as specified in the definition of the partner LU.

partner_lu_def_detail.overlay_size
The size of the returned `partner_lu_def_detail` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `partner_lu_def_detail` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

partner_lu_def_detail.plu_alias
Partner LU alias. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes.

partner_lu_def_detail.fqplu_name
Fully qualified network name for the partner LU. This name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of 1-8 A-string characters, an EBCDIC dot (period) character, and a network name of 1-8 A-string characters.

partner_lu_def_detail.plu_chars.fqplu_name
Fully qualified network name for the partner LU. This name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of 1-8 A-string characters, an EBCDIC dot (period) character, and a network name of 1-8 A-string characters.

partner_lu_def_detail.plu_chars.plu_alias
Partner LU alias. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes.

partner_lu_def_detail.plu_chars.description
A null-terminated text string describing the partner LU, as specified in the definition of the partner LU.

partner_lu_def_detail.plu_chars.plu_un_name
Uninterpreted name of the partner LU. This is an 8-byte type-A EBCDIC string, padded on the right with spaces if the name is shorter than 8 characters.

partner_lu_def_detail.plu_chars.preference
This parameter is reserved.

partner_lu_def_detail.plu_chars.max_mc_ll_send_size
Maximum logical record length, in bytes, that can be sent to the partner LU. This may be in the range 1-32 , 767, or zero to indicate no limit (in which case the maximum is 32,767). Data records that are larger than this are broken down into several LL records before being sent to the partner LU.

partner_lu_def_detail.plu_chars.conv_security_ver
Specifies whether the partner LU is authorized to validate user IDs on behalf of local LUs; that is, whether the partner LU may set the already-verified indicator in an Attach request. Possible values are:

AP_YES
The partner LU is authorized to validate user IDs.

AP_NO
The partner LU is authorized to validate user IDs.

partner_lu_def_detail.plu_chars.parallel_sess_supp
Specifies whether parallel sessions are supported. Possible values are:

AP_YES
Parallel sessions are supported.

AP_NO
Parallel sessions are not supported.

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

ファイル名の変更
list_options パラメーターは、指定された名前から始まるすべての項目をリストするために **リストを含む** （ **包括的** ）に設定されましたが、 *plu_alias* パラメーターまたは *fqplu_name* パラメーターが無効でした。

ファイルの追加リスト・ オプション
list_options パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

QUERY_PORT

QUERY_PORT returns a list of information about a node's ports. This information is structured as "determined data" (data gathered dynamically during execution) and "defined data" (the data supplied by the application on DEFINE_PORT).

This verb can be used to obtain either summary or detailed information, about a specific port or about multiple ports, depending on the options used.

VCB 構造体

```
typedef struct query_port
{
  AP_UINT16        opcode;                /* verb operation code          */
  unsigned char    reserv2;               /* reserved                     */
  unsigned char    format;                /* reserved                     */
  AP_UINT16        primary_rc;            /* primary return code          */
  AP_UINT32        secondary_rc;          /* secondary return code        */
  unsigned char    *buf_ptr;              /* pointer to buffer            */
  AP_UINT32        buf_size;              /* buffer size                  */
  AP_UINT32        total_buf_size;        /* total buffer size required   */
  AP_UINT16        num_entries;           /* number of entries            */
  AP_UINT16        total_num_entries;     /* total number of entries      */
  unsigned char    list_options;          /* listing options              */
  unsigned char    reserv3;               /* reserved                     */
  unsigned char    port_name[8];          /* port name                    */
  unsigned char    dlc_name[8];           /* DLC name filter              */
} QUERY_PORT;
```

```
typedef struct port_summary
{
  AP_UINT16        overlay_size;           /* size of returned entry       */
  unsigned char    port_name[8];           /* port name                    */
  unsigned char    description[32];        /* resource description         */
  unsigned char    reserv2[16];            /* reserved                     */
  unsigned char    port_state;             /* port state                   */
  unsigned char    reserv1[1];             /* reserved                     */
  unsigned char    dlc_name[8];            /* name of DLC                  */
} PORT_SUMMARY;
```

```
typedef struct port_detail
{
  AP_UINT16        overlay_size;           /* size of returned entry       */
  unsigned char    port_name[8];           /* port name                    */
  unsigned char    reserv1[2];             /* reserved                     */
  PORT_DET_DATA    det_data;               /* determined data              */
  PORT_DEF_DATA    def_data;               /* defined data                 */
} PORT_DETAIL;
```

```
typedef struct port_det_data
{
  unsigned char    port_state;             /* port state                   */
  unsigned char    dlc_type;               /* DLC type                     */
  unsigned char    port_sim_rim;           /* port initialization options  */
  unsigned char    reserv1;                /* reserved                     */
  AP_UINT16        def_ls_good_xids;       /* number of successful XIDs    */
  AP_UINT16        def_ls_bad_xids;        /* number of unsuccessful XIDs  */
  AP_UINT16        dyn_ls_good_xids;       /* successful XIDs on dynamic   */
                                           /*  LS count                    */
  AP_UINT16        dyn_ls_bad_xids;        /* failed XIDs on dynamic LS    */
                                           /*  count                       */
  AP_UINT16        num_implicit_links;     /* number of implicit links     */
  unsigned char    neg_ls_supp;            /* negotiable?                  */
  unsigned char    abm_ls_supp;            /* ABM support?                 */
  AP_UINT32        start_time;             /* Start time of the port       */
  unsigned char    reserva[12];            /* reserved                     */
} PORT_DET_DATA;
```

```
typedef struct port_def_data
{
  unsigned char    description[32];         /* resource description          */
  unsigned char    initially_active;        /* is the port initially active? */
  unsigned char    reserv2[15];             /* reserved                      */
  unsigned char    dlc_name[8];             /* DLC name associated with port */
  unsigned char    port_type;               /* port type                     */
  unsigned char    port_attributes[4];      /* port attributes               */
  unsigned char    implicit_uplink_to_en;   /* implicit EN links up or down? */
  unsigned char    implicit_appn_links_len; /* reserved                      */
  unsigned char    reserv3;                 /* reserved                      */
  AP_UINT32        port_number;             /* port number                   */
  AP_UINT16        max_rcv_btu_size;        /* max receive BTU size          */
  AP_UINT16        tot_link_act_lim;        /* total link activation limit   */
  AP_UINT16        inb_link_act_lim;        /* inbound link activation limit */
  AP_UINT16        out_link_act_lim;        /* outbound link activation limit*/
  unsigned char    ls_role;                 /* initial link station role     */
  unsigned char    retry_flags;             /* reserved                      */
  AP_UINT16        max_activation_attempts; /* reserved                      */
  AP_UINT16        activation_delay_timer;  /* reserved                      */
  unsigned char    mltg_pacing_algorithm;   /* reserved                      */
  unsigned char    implicit_tg_sharing_prohibited; /* reserved               */
  unsigned char    link_spec_data_format;   /* reserved                      */
  unsigned char    limit_enable;            /* reserved                      */
  unsigned char    reserv1[6];              /* reserved                      */
  unsigned char    implicit_dspu_template[8];      /* implicit dspu template */
  AP_UINT16        implicit_ls_limit;       /* implicit ls limit             */
  unsigned char    reserv4;                 /* reserved                      */
  unsigned char    implicit_dspu_services;  /* reserved                      */
  unsigned char    implicit_deact_timer;    /* deact timer for implicit LSs  */
  AP_UINT16        act_xid_exchange_limit;  /* activation XID exchange limit */
  AP_UINT16        nonact_xid_exchange_limit;     /* non-act. XID exchange   */
                                                  /*  limit                  */
  unsigned char    ls_xmit_rcv_cap;         /* LS transmit-receive capability*/
  unsigned char    max_ifrm_rcvd;           /* maximum number of I-frames    */
                                            /*  that can be received         */
  AP_UINT16        target_pacing_count;     /* target pacing count           */
  AP_UINT16        max_send_btu_size;       /* maximum send BTU size         */
  LINK_ADDRESS     dlc_data;                /* DLC data                      */
```

```
   LINK_ADDRESS     hpr_dlc_data;            /* reserved                      */
   unsigned char    implicit_cp_cp_sess_support;   /* implicit links allow    */
                                                   /*  CP-CP sessions         */
   unsigned char    implicit_limited_resource;     /* implicit links are      */
                                                   /*  limited resource       */
   unsigned char   implicit_hpr_support;     /* Implicit links support HPR    */
   unsigned char   implicit_link_lvl_error;  /* Send HPR traffic on implicit  */
                                             /*  links using link-level error */
                                             /*  recovery?                    */
   unsigned char   retired1;                 /* reserved                      */
   TG_DEFINED_CHARS default_tg_chars;        /* default TG chars              */
   unsigned char   discovery_supported;      /* reserved                      */
   AP_UINT16       port_spec_data_len;       /* length of port specific data  */
   AP_UINT16       link_spec_data_len;       /* length of link specific data  */
} PORT_DEF_DATA;
```

For more details of the リンク・アドレス structure, see 370 ページの『QUERY_LS』; for more details of the port-specific and link-specific data, see 158 ページの『DEFINE_PORT』 and 104 ページの『DEFINE_LS』. ポート固有データのデータ構造は、 `port_def_data` 構造体の後に続き、リンク固有データのデータ構造はこれに従います。両方の構造体は、 4 バイト境界で開始するために埋め込まれます。

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
AP_QUERY_PORT

buf_ptr
CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

num_entries
データが戻されるポートの最大数。 特定の範囲ではなく、特定のポートのデータを要求するには、 1 という値を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、 CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータの戻りを開始する必要のあるリスト内の位置、および各項目に必要な情報のレベル。 以下の値のいずれかを使用して、情報のレベルを指定します。

要約の要約
サマリー情報のみ。

追加の詳細
詳細情報。

以下のいずれかの値を指定して、論理 **それとも** 操作を使用してこの値を結合します。

リストの最初のリスト （ _R ）
リストの最初の項目から開始します。

リストを含む （ 包括的 ）
ポート名 パラメーターで指定されたエントリーから開始します。

次への AP_LIST_FROM_NEXT
ポート名 パラメーターによって指定されたエントリーの直後のエントリーから開始します。

リストの順序とアプリケーションでの特定のエントリーの取得方法について詳しくは、 34 ページの『List options for QUERY_* Verbs』を参照してください。

ポート名
照会されるポートの名前。 これは 8 バイトの ASCII ストリングで、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。 *list_options* を **リストの最初のリスト** （ _R ）に設定すると、このパラメーターは無視されます。

dlc_name
DLC 名フィルター。 特定の DLC に関連付けられたポートに関する情報のみを戻すには、 DLC 名を指定します。 これは 8 バイトの ASCII ストリングで、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。 DLC 名にフィルターを適用せずにすべてのポートに関する情報を戻すには、このパラメーターを 8 進ゼロに設定します。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

port_summary.overlay_size
The size of the returned `port_summary` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `port_summary` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

port_summary.port_name
Name of the port. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes.

port_summary.description
A null-terminated text string describing the port, as specified in the definition of the port.

port_summary.port_state
Specifies the current state of the port. Possible values are:

AP_ACTIVE
The port is active.

AP_NOT_ACTIVE
The port is not active.

AP_PENDING_ACTIVE
START_PORT is in progress.

AP_PENDING_INACTIVE
STOP_PORT is in progress.

port_summary.dlc_name
Name of the DLC associated with this port. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes.

port_detail.overlay_size
The size of the returned `port_detail` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `port_detail` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

port_detail.port_name
Name of the port. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes.

port_detail.det_data.port_state
Specifies the current state of the port. Possible values are:

AP_ACTIVE
The port is active.

AP_NOT_ACTIVE
The port is not active.

AP_PENDING_ACTIVE
START_PORT is in progress.

AP_PENDING_INACTIVE
STOP_PORT is in progress.

port_detail.det_data.dlc_type
DLC type for the port. This is one of the following:

AP_SDLC
SDLC

AP_X25
QLLC

AP_TR
Token Ring

AP_ETHERNET
Ethernet

AP_MPC
Multipath Channel (MPC), CS Linux for IBM Z only

AP_IP
Enterprise Extender (HPR/IP)

port_detail.det_data.port_sim_rim
Specifies whether Set Initialization Mode (SIM) and Receive Initialization Mode (RIM) are supported. Possible values are:

AP_YES
SIM and RIM are supported.

AP_NO
SIM and RIM are not supported.

port_detail.det_data.def_ls_good_xids
Total number of successful XID exchanges that have occurred on all defined link stations on this port since the last time this port was started.

port_detail.det_data.def_ls_bad_xids
Total number of unsuccessful XID exchanges that have occurred on all defined link stations on this port since the last time this port was started.

port_detail.det_data.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.

port_detail.det_data.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.

port_detail.det_data.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 of `port_def_data`.

port_detail.det_data.neg_ls_supp
Support for negotiable link stations. Possible values are:

AP_YES
Link stations can be negotiated.

AP_NO
Link stations cannot be negotiated.

port_detail.det_data.abm_ls_supp
Support for ABM link stations. Possible values are:

AP_YES
ABM link stations are supported.

AP_NO
ABM link stations are not supported.

AP_UNKNOWN
Support for ABM link stations cannot be determined because the DLC associated with this port has not yet been started.

port_detail.det_data.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.

port_detail.def_data.description
A null-terminated text string describing the port, as specified in the definition of the port.

port_detail.def_data.dlc_name
Name of the DLC associated with this port. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes.

port_detail.def_data.port_type
The type of line used by the port.

For SDLC, the following values may be returned:

AP_PORT_SWITCHED
Switched line.

AP_PORT_NONSWITCHED
Nonswitched line.

For QLLC, this is set to `AP_PORT_SWITCHED`.

For Token Ring / Ethernet, this is set to `AP_PORT_SATF` (shared access transport facility).

For Enterprise Extender (HPR/IP), this is set to `AP_PORT_SATF` (shared access transport facility).

port_detail.def_data.port_attributes
This is a bit field. It can take the value `AP_NO`, or the following:

AP_RESOLVE_BY_LINK_ADDRESS
This value specifies that an attempt is made to resolve incoming calls by using the link address on CONNECT_IN before using the CP name (or node ID) carried on the received XID3 to resolve them. This is ignored if the *port_type* parameter is not set to `AP_PORT_SWITCHED`.

def_data.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:

AP_YES
Implicit links to an End Node are uplinks.

AP_NO
Implicit links to an End Node are downlinks.

port_detail.def_data.port_number
Port number.

port_detail.def_data.max_rcv_btu_size
Maximum BTU size that can be received.

port_detail.def_data.tot_link_act_lim
Total link activation limit.

port_detail.def_data.inb_link_act_lim
Inbound link activation limit.

port_detail.def_data.out_link_act_lim
Outbound link activation limit.

port_detail.def_data.ls_role
Link station role.

For SDLC or QLLC, the following values may be returned:

AP_LS_PRI
Primary

AP_LS_SEC
Secondary

AP_LS_NEG
Negotiable

For Token Ring / Ethernet, this is set to AP_LS_NEG (negotiable).

port_detail.def_data.implicit_dspu_template
Specifies the DSPU template, defined with the DEFINE_DSPU_TEMPLATE verb, that will be 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 is an 8-byte string in a locally displayable character set. All eight bytes are significant and must be set.

If the *def_data.implicit_dspu_services* parameter is not set to AP_PU_CONCENTRATION, this parameter is reserved.

port_detail.def_data.implicit_ls_limit
The maximum number of implicit link stations which can be active on this port simultaneously, including dynamic links and links activated for Discovery. A value of zero indicates that there is no limit; a value of AP_NO_IMPLICIT_LINKS indicates that no implicit links are allowed.

port_detail.def_data.implicit_deact_timer
Limited resource link deactivation timer, in seconds.

If *implicit_limited_resource* is set to AP_YES or AP_NO_SESSIONS, then an HPR-capable implicit link is automatically deactivated if no data flows on the link for the duration of this timer, and no sessions are using the link.

If *implicit_limited_resource* is set to AP_INACTIVITY, then an implicit link is automatically deactivated if no data flows on the link for the duration of this timer.

port_detail.def_data.act_xid_exchange_limit
Activation XID exchange limit.

port_detail.def_data.nonact_xid_exchange_limit
Non-activation XID exchange limit.

port_detail.def_data.ls_xmit_rcv_cap
Specifies the link station transmit/receive capability. Possible values are:

AP_LS_TWS
Two-way simultaneous

AP_LS_TWA
Two-way alternating

port_detail.def_data.max_ifrm_rcvd
Maximum number of I-frames that can be received by local link stations before an acknowledgment is sent. Range: 1-127.

port_detail.def_data.target_pacing_count
Numeric value between 1 and 32,767 inclusive indicating the desired pacing window size. (The current version of CS Linux does not make use of this value.)

port_detail.def_data.max_send_btu_size
Maximum BTU size that can be sent.

port_detail.def_data.dlc_data
Port address. For more information on the `dlc_data` structure, see “QUERY_LS” on page 370.

def_data.implicit_cp_cp_sess_support
Specifies whether CP-CP sessions are permitted for implicit link stations using this port. Possible values are:

AP_YES
CP-CP sessions are permitted for implicit LSs.

AP_NO
CP-CP sessions are not permitted for implicit LSs.

def_data.implicit_limited_resource
Specifies whether implicit link stations off this port are defined as limited resources. Possible values are:

AP_NO
Implicit links are not limited resources, and will not be deactivated automatically.

AP_NO_SESSIONS
Implicit links are limited resources, and will be deactivated automatically when no active sessions are using them.

AP_INACTIVITY
Implicit links are limited resources, and will be deactivated automatically when no active sessions are using them or when no data has flowed for the time period specified by the *implicit_deact_timer* field.

def_data.implicit_hpr_support
Specifies whether High Performance Routing (HPR) is supported on implicit links. Possible values are:

AP_YES
HPR is supported on implicit links.

AP_NO
HPR is not supported on implicit links.

def_data.implicit_link_lvl_error
For SDLC, this parameter is not used.

For other link types, this parameter specifies whether HPR traffic should be sent on implicit links using link-level error recovery (AP_YES or AP_NO). The parameter is reserved if *implicit_hpr_support* is set to AP_NO.

def_data.default_tg_chars
Default TG characteristics. These are used for implicit link stations using this port, and as the default TG characteristics for defined link stations that do not have TG characteristics explicitly defined. For details of these parameters, see “DEFINE_LS” on page 104.

port_detail.def_data.port_spec_data_len
Unpadded length, in bytes, of the port-specific data. The data structure for this data follows the `port_def_data` structure, but is padded to start on a 4-byte boundary. For more details of the port-specific data, see “DEFINE_PORT” on page 158.

port_detail.def_data.link_spec_data_len
Unpadded length, in bytes, of the link-specific data. The data structure for the link-specific data follows the data structure for the port-specific data, but is padded to start on a 4-byte boundary. For more details of the link-specific data, see “DEFINE_PORT” on page 158.

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_PORT_NAME
The *list_options* parameter was set to AP_LIST_INCLUSIVE to list all entries starting from the supplied name, but the *port_name* parameter was not valid.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

クエリー・プ

QUERY_PU は、ローカル PU およびそれに関連したリンクに関する情報を戻します。 この verb を使用して、使用するオプションに応じて、特定の PU または複数の PU に関する情報を取得することができます。

VCB 構造体

```
typedef struct query_pu
{
   AP_UINT16       opcode;               /* verb operation code            */
   unsigned char   reserv2;              /* reserved                       */
   unsigned char   format;               /* reserved                       */
   AP_UINT16       primary_rc;           /* primary return code            */
   AP_UINT32       secondary_rc;         /* secondary return code          */
   unsigned char   *buf_ptr;             /* pointer to buffer              */
   AP_UINT32       buf_size;             /* buffer size                    */
   AP_UINT32       total_buf_size;       /* total buffer size required     */
   AP_UINT16       num_entries;          /* number of entries              */
   AP_UINT16       total_num_entries;    /* total number of entries        */
   unsigned char   list_options;         /* listing options                */
   unsigned char   reserv3;              /* reserved                       */
   unsigned char   pu_name[8];           /* PU name                        */
   unsigned char   host_attachment;      /* host attachment filter         */
} QUERY_PU;
```

```
typedef struct pu_data
{
   AP_UINT16       overlay_size;          /* size of returned entry          */
   unsigned char   pu_name[8];            /* PU name                         */
   unsigned char   description[32];       /* resource description            */
   unsigned char   reserv1[16];           /* reserved                        */
   unsigned char   ls_name[8];            /* LS name                         */
   unsigned char   pu_sscp_sess_active;   /* Is PU-SSCP session active       */
   unsigned char   host_attachment;       /* Host attachment                 */
   SESSION_STATS   pu_sscp_stats;         /* PU-SSCP session statistics      */
   unsigned char   sscp_id[6];            /* SSCP ID                         */
   unsigned char   conventional_lu_compression; /* compression for LU 0-3?   */
   unsigned char   conventional_lu_cryptography;   /* reserved               */
   unsigned char   dddlu_supported;       /* does the host support DDDLU?    */
   unsigned char   tcpcv_supported;       /* does the host support TCPCVs?   */
   unsigned char   dddlu_offline_supported;  /* does the PU support sending  */
                                          /*  NMVT (power off) to host?   */
   unsigned char   reserva[9];            /* reserved                        */
} PU_DATA;
```

```
typedef struct session_stats
{
   AP_UINT16       rcv_ru_size;           /* session receive RU size         */
   AP_UINT16       send_ru_size;          /* session send RU size            */
   AP_UINT16       max_send_btu_size;     /* maximum send BTU size           */
   AP_UINT16       max_rcv_btu_size;      /* maximum rcv BTU size            */
   AP_UINT16       max_send_pac_win;      /* maximum send pacing window size */
   AP_UINT16       cur_send_pac_win;      /* current send pacing window size */
   AP_UINT16       max_rcv_pac_win;       /* maximum receive pacing window   */
                                          /*  size                           */
   AP_UINT16       cur_rcv_pac_win;       /* current receive pacing window   */
                                          /*  size                           */
   AP_UINT32       send_data_frames;      /* number of data frames sent      */
   AP_UINT32       send_fmd_data_frames;  /* num fmd data frames sent        */
   AP_UINT32       send_data_bytes;       /* number of data bytes sent       */
   AP_UINT32       rcv_data_frames;       /* number of data frames received  */
   AP_UINT32       rcv_fmd_data_frames;   /* num fmd data frames received    */
   AP_UINT32       rcv_data_bytes;        /* number of data bytes received   */
   unsigned char   sidh;                  /* session ID high byte (from      */
                                          /*  LFSID)                         */
   unsigned char   sidl;                  /* session ID low byte (from LFSID)*/
   unsigned char   odai;                  /* ODAI bit set                    */
   unsigned char   ls_name[8];            /* Link station name               */
   unsigned char   pacing_type;           /* type of pacing in use           */
} SESSION_STATS;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
AP_QUERY_PU

buf_ptr
CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

num_entries
データが戻される PU の最大数。 範囲ではなく、特定の PU のデータを要求するには、 1 という値を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、 CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータの戻りを開始する必要があるリスト内の位置。 次の値のいずれかを指定してください。

リストの最初のリスト （ _R ）
リストの最初の項目から開始します。

リストを含む （ 包括的 ）
プール名 パラメーターで指定されたエントリーから開始します。

次への AP_LIST_FROM_NEXT
プール名 パラメーターによって指定されたエントリーの直後のエントリーから開始します。

リストの順序とアプリケーションでの特定のエントリーの取得方法について詳しくは、34 ページの『List options for QUERY_* Verbs』を参照してください。

プール名
情報が必要な PU の名前、または PU のリストの索引として使用される名前。 *list_options* を **リストの最初のリスト** (_R)に設定すると、この値は無視されます。 これは 8 バイトのタイプ A の EBCDIC ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

ホスト接続
PU がホストに直接接続されているか、または DLUR を使用して、戻された情報をフィルターに掛けるかどうかを指定します。 可能な値は次のとおりです

追加指示が付加されます
ホスト・システムに直接接続されている PU に関する情報のみを戻します。

付加された付加
DLUR によってサポートされる PU に関する情報のみを戻します。

追加なし
ホスト接続機構に関係なく、すべての PU に関する情報を戻します。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

pu_data.overlay_size
The size of the returned `pu_data` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `pu_data` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

pu_data.pu_name
PU Name. This is an 8-byte type-A EBCDIC string, padded on the right with spaces if the name is shorter than 8 characters.

pu_data.description
A null-terminated text string describing the PU, as specified in the definition of the LS or of the internal PU.

pu_data.ls_name
Name of the link station associated with this PU. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes.

pu_data.pu_sscp_sess_active
Specifies whether the PU-SSCP session is active. Possible values are:

AP_YES
The PU-SSCP session is active.

AP_NO
The PU-SSCP session is inactive.

pu_data.host_attachment
Local PU host attachment type.

Possible values are:

AP_DIRECT_ATTACHED
PU is directly attached to the host system.

AP_DLUR_ATTACHED
PU is supported by DLUR.

pu_data.pu_sscp_stats.rcv_ru_size
Reserved (always set to zero).

pu_data.pu_sscp_stats.send_ru_size
Reserved (always set to zero).

pu_data.pu_sscp_stats.max_send_btu_size
Maximum BTU size that can be sent.

pu_data.pu_sscp_stats.max_rcv_btu_size
Maximum BTU size that can be received.

pu_data.pu_sscp_stats.max_send_pac_win
Reserved (always set to zero).

pu_data.pu_sscp_stats.cur_send_pac_win
Reserved (always set to zero).

pu_data.pu_sscp_stats.max_rcv_pac_win
Reserved (always set to zero).

pu_data.pu_sscp_stats.cur_rcv_pac_win
Reserved (always set to zero).

pu_data.pu_sscp_stats.send_data_frames
Number of normal flow data frames sent.

pu_data.pu_sscp_stats.send_fmd_data_frames
Number of normal flow FMD data frames sent.

pu_data.pu_sscp_stats.send_data_bytes
Number of normal flow data bytes sent.

pu_data.pu_sscp_stats.rcv_data_frames
Number of normal flow data frames received.

ppu_data.pu_sscp_stats.rcv_fmd_data_frames
Number of normal flow FMD data frames received.

pu_data.pu_sscp_stats.rcv_data_bytes
Number of normal flow data bytes received.

pu_data.pu_sscp_stats.sidh
Session ID high byte.

pu_data.pu_sscp_stats.sidl
Session ID low byte.

pu_data.pu_sscp_stats.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.

pu_data.pu_sscp_stats.ls_name
Link station name associated with statistics. This is an 8-byte ASCII character string, right-padded with spaces if the name is shorter than 8 characters.

pu_data.pu_sscp_stats.pacing_type
The type of receive pacing in use on the PU-SSCP session. This parameter is set to AP_NONE.

pu_data.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.

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

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

AP_NO
Data compression should not be used for LU 0-3 sessions using this PU.

pu_data.dddlu_supported
Specifies whether the host system supports DDDLU (Dynamic Definition of Dependent LUs). Possible values are:

AP_YES
The host supports DDDLU.

AP_NO
The host does not support DDDLU.

pu_data.tcpcv_supported
Specifies whether the host system supports receiving the TCP/IP Information Control Vector (0x64). CS Linux can use this vector to send TCP/IP addressing information for TN3270 or LUA clients to the host. Possible values are:

AP_YES
The host supports TCP CVs.

AP_NO
The host does not support TCP CVs.

pu_data.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), CS 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:

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

AP_NO
The local PU does not send NMVT (power off) messages to the host.

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_PU_NAME
The *list_options* parameter was set to AP_LIST_INCLUSIVE to list all entries starting from the supplied name, but the *pu_name* parameter was not valid.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: state check

If the verb does not execute because of a state error, CS Linux returns the following parameters.

primary_rc
AP_STATE_CHECK

secondary_rc

AP_INVALID_PU_TYPE
The PU specified by the *pu_name* parameter is a remote PU and not a local PU.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_STATE_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

照会 RAPI_受給者

QUERY_RAPI_CLIENTS は、リモート API クライアント (AIX、 Linux、または Windows 上) に関する情報を戻します。これは、 CS Linux LAN 上の特定のサーバーが現在コントローラーとして機能しているものです。

この verb はサーバーに対して発行する必要があります。 ノードがそのサーバー上で開始されているかどうかは関係ありません。

注 : クライアントが Web サーバー経由でサーバーに接続されており、クライアント・ソフトウェアが停止している場合、 Web サーバーが CS Linux コントローラー・サーバーへの接続を終了する前に、 1 分か 2 分の遅延が発生することがあります。 これは、サーバーの使用を停止した後も、 QUERY_RAPI_CLIENTS verb に短時間クライアントが含まれている可能性があることを意味します。

VCB structure

```
typedef struct query_rapi_clients
{
   AP_UINT16              opcode;            /* verb operation code            */
   unsigned char          reserv2;           /* reserved                       */
   unsigned char          format;            /* reserved                       */
   AP_UINT16              primary_rc;        /* primary return code            */
   AP_UINT32              secondary_rc;      /* secondary return code          */
   unsigned char          *buf_ptr;          /* pointer to buffer              */
   AP_UINT32              buf_size;          /* buffer size                    */
   AP_UINT32              total_buf_size;    /* total buffer size required     */
   AP_UINT16              num_entries;       /* number of entries              */
   AP_UINT16              total_num_entries; /* total number of entries        */
   unsigned char          list_options;      /* listing options                */
   AP_UINT16              max_clients;       /* maximum number of clients      */
   unsigned char          sys_name[128];     /* RAPI Client to start query     */
} QUERY_RAPI_CLIENTS;
```

```
typedef struct rapi_client_info
{
   AP_UINT16        overlay_size;             /* overlay size                 */
   unsigned char    sys_name[128];            /* RAPI Client System name      */
   SNA_IP_ADDR      rapi_client_origin_ip_addr;  /* IP addr client sends us   */
   SNA_IP_ADDR      rapi_client_adj_ip_addr;  /* IP addr client comes in on   */
   AP_UINT16        rapi_client_adj_port;     /* port IP client comes in on   */
} RAPI_CLIENT_INFO;
```

```
typedef struct sna_ip_addr
{
   AP_UINT16        family;                   /* IPv4 or IPv6                 */
   union
   {
     unsigned char  ipv4_addr[4];
     unsigned char  ipv6_addr[16];
   } ip_addr;
} SNA_IP_ADDR;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
AP_QUERY_RAPI_受給者

buf_ptr
CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

num_entries
データが戻されるクライアントの最大数。 特定の範囲ではなく、特定のクライアントのデータを要求するには、 1 という値を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、 CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータの戻りを開始する必要があるリスト内の位置。 可能な値は次のとおりです

リストの最初のリスト （ _R ）
クライアントのリストの最初の項目から開始します。

リストを含む （ 包括的 ）
システム名 パラメーターで指定されたエントリーから開始します。

次への AP_LIST_FROM_NEXT
システム名 パラメーターによって指定されたエントリーの直後のエントリーから開始します。

システム名
リスト内の索引として使用されるクライアントの完全修飾システム名 (newbox.this.co.uk など) 。 *list_options* を **リストの最初のリスト** （ _R ）に設定すると、このパラメーターは無視されます。

これは 1 文字から 128 文字の ASCII ストリングで、名前が 128 文字より短い場合は右側にスペースが埋め込まれます。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

buf_size
提供されたバッファーに戻された情報の長さ。

total_buf_size
要求されたすべてのリスト情報を戻すために必要となるバッファーのサイズを示す戻り値。 *buf_size* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

num_entries
データ・バッファーに戻されたエントリーの数。

total_num_entries
使用可能な項目の合計数。 *num_entries* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

データ・バッファー内の各項目は、以下のパラメーターで構成されます。

クライアントの最大数
CS Linux ソフトウェアの開始以降、任意の時点でサーバーをコントローラー・サーバーとして使用しているクライアントの最大数。

***rapi_client_info.*オーバーレイ・サイズ**
戻された `client_info` **をレイプ** 構造体のサイズ。すなわち、データ・バッファー内の次のエントリーの先頭までのオフセット。

アプリケーションが戻されたバッファーを調べて、各 `client_info` **をレイプ** 構造体を順番に検出する必要がある場合は、この値を使用して次のデータ構造の正しいオフセットに移動しなければなりません。また、 C `sizeof()` 演算子を使用してはなりません。 これは、返されるオーバーレイのサイズが、 CS Linux の将来のリリースで増加する可能性があるためです。戻されたオーバーレイ・サイズを使用すると、アプリケーションが今後のリリースで引き続き動作することが保証されます。

***client_client_info.sys_name* のレイプ**
クライアントの完全修飾システム名 (`newbox.this.co.uk` など) 。

***client_origin* 性 *rapi_client_origin_ip_addr* によるレイプされた**
クライアントの IP アドレス。

***client_origin_client_origin_ip_addr.*ファミリーをレイプしました**
クライアントに指定された TCP/IP アドレスのタイプ。 使用可能な値は以下のとおりです。

アフリネット
IPv4 アドレス。小数点付き 10 進数アドレス (`193.1.11.100` など) として指定されます。

ネット受信 6
IPv6 アドレス。コロン付き 16 進アドレス (`2001:0db8:0000:0000:0000:0000:1428:57ab` や `2001:db8::1428:57ab` など) として指定します。

注 : 値 **アフリネット** と **ネット受信 6** は、システム・ヘッダー・ファイルから取得され、 CS Linux によって定義された標準の **追加 *** 値ではありません。 システム・ヘッダー・ファイルは `/usr/include/linux/socket.h` は、 Linux サーバーまたはクライアント上で、 `/usr/include/sys/socket.h` は AIX クライアント上にあります。 です。

If your NOF application needs to test against these values, you should use `#include` to include this system file in addition to the **ノード・ キュー・ ファイル** ヘッダー・ファイル。

***client_client_origin_addr.ip_addr.ipv4_addr* にレイプされました。**
このフィールドは、 家族 パラメーターが **アフリネット**に設定されている場合にのみ使用されます。クライアント・コンピューターの IPv4 (ドット 10 進) アドレス。

***client_client_origin_addr.ip_addr.ipv6_addr* にレイプされました。**
このフィールドは、 家族 パラメーターが **ネット受信 6** に設定されている場合にのみ使用されます。クライアント・コンピューターの IPv6 (コロン 16 進) アドレス。

***client_client_adj_ip_addr* をレイプ *_client_adj_addr* に**
クライアントが CS Linux に接続するときに使用する IP アドレス。 これは、以下のいずれかが真の場合には クライアントの発信元 *_ip_addr* と同じではありません。

- クライアントは Web サーバーを介して接続します。
- クライアントは、 TCP/IP プロキシーまたは NAT ルーター(Linux iptables ツールなど) を介して接続します。

- クライアントに複数の IP アドレスがあります。

レイプ・*client_info.rapi_client_adj_addr.*ファミリー
クライアントが CS Linux に接続するときに使用する TCP/IP アドレスのタイプ。 使用可能な値は以下のとおりです。

アフリネット
IPv4 アドレス。小数点付き 10 進数アドレス (`193.1.11.100` など) として指定されます。

ネット受信 6
IPv6 アドレス。コロン付き 16 進アドレス (`2001:0db8:0000:0000:0000:0000:0000:1428:57ab` や `2001:db8::1428:57ab` など) として指定します。

注 : 値 **アフリネット** と **ネット受信 6** は、システム・ヘッダー・ファイルから取得され、 CS Linux によって定義された標準の **追加 *** 値ではありません。 システム・ヘッダー・ファイルは `/usr/include/linux/socket.h` は、 Linux サーバーまたはクライアント上で、 `/usr/include/sys/socket.h` は AIX クライアント上にあります。 です。

If your NOF application needs to test against these values, you should use `#include` to include this system file in addition to the **ノード· キュー· ファイル** ヘッダー・ファイル。

***client_client_info.ip_client_adj_addr.ipv4_addr* にレイプされました。**
このフィールドは、 家族 パラメーターが **アフリネット**に設定されている場合にのみ使用されます。 クライアントが CS Linux に接続するときに使用する IPv4 (ドット 10 進) アドレス。

***client_client_adj_ip_addr.ipv6_addr* にレイプされました。**
このフィールドは、 家族 パラメーターが **ネット受信 6** に設定されている場合にのみ使用されます。 クライアントが CS Linux に接続するために使用する IPv6 (コロン 16 進数) アドレス。

ラピ*_client_info.rapi_client_adj_port*
クライアントが CS Linux に接続するときに使用する IP ポート番号。

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

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

AP_INVALID_NODE_NAME
The *list_options* parameter was set to AP_LIST_INCLUSIVE or AP_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.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

QUERY_RCF_ACCESS

QUERY_RCF_ACCESS returns information about the permitted access to the CS Linux Remote Command Facility (RCF): the user ID used to run UNIX Command Facility (UCF) commands, and the restrictions on which administration commands can be issued using the Service Point Command Facility (SPCF). This

information was previously set up using DEFINE_RCF_ACCESS. For more information about SPCF and UCF, see the *IBM Communications Server for Data Center Deployment on Linux Administration Guide*.

This verb must be issued to the domain configuration file.

VCB 構造体

```
typedef struct query_rcf_access
{
   AP_UINT16         opcode;                  /* Verb operation code      */
   unsigned char     reserv2;                 /* reserved                 */
   unsigned char     format;                  /* reserved                 */
   AP_UINT16         primary_rc;              /* primary return code      */
   AP_UINT32         secondary_rc;            /* secondary return code    */
   unsigned char     ucf_username[32];        /* UCF username             */
   AP_UINT32         spcf_permissions;        /* SPCF permissions         */
   unsigned char     reserv3[8];              /* Reserved                 */
} QUERY_RCF_ACCESS;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_RCF_ACCESS

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

***ucf_* ユーザー名**
UCF ユーザーの Linux ユーザー名を指定します。 このパラメーターは、ヌル終了 ASCII ストリングです。

すべての UCF コマンドは、このユーザーのユーザー ID を使用して実行されます。このユーザーに対して定義されたデフォルト・シェルおよびアクセス許可を使用します。

このパラメーターがヌル・ストリングに設定されている場合は、 UCF アクセスが禁止されていることを示します。

***spcf_*許可**
SPCF を使用してアクセスできる CS Linux 管理コマンドのタイプを指定します。 これは、 SPCF アクセスが禁止されていることを示すため、または以下の値の 1 つ以上 (論理 **それとも**を使用して結合される) を示すために、 **追加なし** に設定されます。

カタログを許可するローカル
QUERY_* verb は許可されています。

ローカルで許可さないローカル
DEFINE_*、 SET_*、 DELETE_*、 ADD_*、および REMOVE_* verb、および INIT_NODE も許可されています。

アプリケーション・アクション・ローカル (_R)
"アクション" の verb は許可されます。 START_*、 STOP_*、 ACTIVATE_*、 DEACTIVATE_*、および、 APING、 INITIALIZE_SESSION_LIMIT、 CHANGE_SESSION_LIMIT、および RESET_SESSION_LIMIT があります。

カタログを許可するリモート
QUERY_* verb は、リモート CS Linux ノードへのアクセスを可能にするために使用できます。

許可を許可さない (リモート)
DEFINE_*、 SET_*、 DELETE_*、 ADD_*、 REMOVE_*、および INIT_NODE verb は、リモート CS Linux ノードへのアクセスを許可されています。

追加アクションを実行 （ リモート ）
START_*、 STOP_*、 ACTIVATE_*、 DEACTIVATE_*、 APING、 INITIALIZE_SESSION_LIMIT 、および RESET_SESSION_LIMIT verb は、リモート CS Linux ノードへのアクセスを提供することが許可されています。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

QUERY_RTP_CONNECTION

The QUERY_RTP_CONNECTION verb returns a list of information about Rapid Transport Protocol (RTP) connections for which the node is an endpoint. This verb 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 verb must be issued to a running node.

VCB structure

```
typedef struct query_rtp_connection
{
   AP_UINT16       opcode;              /* verb operation code             */
   unsigned char   reserv2;             /* reserved                        */
   unsigned char   format;              /* reserved                        */
   AP_UINT16       primary_rc;          /* primary return code             */
   AP_UINT32       secondary_rc;        /* secondary return code           */
   unsigned char   *buf_ptr;            /* pointer to buffer               */
   AP_UINT32       buf_size;            /* buffer size                     */
   AP_UINT32       total_buf_size;      /* total buffer size required      */
   AP_UINT16       num_entries;         /* number of entries               */
   AP_UINT16       total_num_entries;   /* total number of entries         */
   unsigned char   list_options;        /* listing options                 */
   unsigned char   reserv3;             /* reserved                        */
   unsigned char   rtp_name[8];         /* name of RTP connection          */
} QUERY_RTP_CONNECTION;
```

```
typedef struct rtp_connection_summary
{
   AP_UINT16       overlay_size;        /* size of returned entry          */
   unsigned char   rtp_name[8];         /* RTP connection name             */
   unsigned char   first_hop_ls_name[8]; /* LS name of first hop           */
   unsigned char   dest_node_name[17];  /* fully qualified name of         */
                                        /*  destination node               */
   unsigned char   connection_type;     /* LU-LU or CP-CP connection?      */
   unsigned char   cos_name[8];         /* class of service name           */
   AP_UINT16       num_sess_active;     /* number of active sessions       */
} RTP_CONNECTION_SUMMARY;
```

```
typedef struct rtp_connection_detail
{
   AP_UINT16       overlay_size;        /* size of returned entry          */
   unsigned char   rtp_name[8];         /* RTP connection name             */
   unsigned char   first_hop_ls_name[8]; /* LS name of first hop           */
   unsigned char   dest_node_name[17];  /* fully qualified name of         */
                                        /*  destination node               */
   unsigned char   isr_boundary_fn;     /* is conn used for Boundary Func? */
   unsigned char   connection_type;     /* LU-LU or CP-CP connection?      */
   unsigned char   reserv1;             /* reserved                        */
   unsigned char   cos_name[8];         /* class of service name           */
   AP_UINT16       max_btu_size;        /* maximum BTU size                */
   AP_UINT32       liveness_timer;      /* liveness timer                  */
   unsigned char   local_tcid[8];       /* local tcid                      */
   unsigned char   remote_tcid[8];      /* remote tcid                     */
   RTP_STATISTICS  rtp_stats;           /* RTP statistics                  */
   AP_UINT16       num_sess_active;     /* number of active sessions       */
   unsigned char   arb_mode;            /* ARB-S, ARB-R, ARB-P?            */
   unsigned char   refifo;              /* refifo support?                 */
   AP_UINT32       refifo_timer;        /* last refifo timer in ms         */
   AP_UINT32       path_switch_time;    /* time since last path switch secs */
```

```
   AP_UINT16       path_switch_atts;     /* number of path switch attempts   */
   unsigned char   reserv2[4];           /* reserved                         */
   AP_UINT16       rscv_len;             /* length of appended RSCV          */
 } RTP_CONNECTION_DETAIL;
```

The session detail structure may be followed by a 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.

```
typedef struct rtp_statistics
{
   AP_UINT32       bytes_sent;           /* total number of bytes sent       */
   AP_UINT32       bytes_received;       /* total number of bytes received   */
   AP_UINT32       bytes_resent;         /* total number of bytes resent     */
   AP_UINT32       bytes_discarded;      /* total number of bytes discarded  */
   AP_UINT32       packets_sent;         /* total number of packets sent     */
   AP_UINT32       packets_received;     /* total number of packets received */
   AP_UINT32       packets_resent;       /* total number of packets resent   */
   AP_UINT32       packets_discarded;    /* total number of packets discarded*/
   AP_UINT32       gaps_detected;        /* gaps detected                    */
   AP_UINT32       send_rate;            /* current send rate                */
   AP_UINT32       max_send_rate;        /* maximum send rate                */
   AP_UINT32       min_send_rate;        /* minimum send rate                */
   AP_UINT32       receive_rate;         /* current send rate                */
   AP_UINT32       max_receive_rate;     /* maximum receive rate             */
   AP_UINT32       min_receive_rate;     /* minimum receive rate             */
   AP_UINT32       burst_size;           /* current burst size               */
   AP_UINT32       up_time;              /* total uptime of connection       */
   AP_UINT32       smooth_rtt;           /* smoothed round-trip time         */
   AP_UINT32       last_rtt;             /* last round-trip time             */
   AP_UINT32       short_req_timer;      /* SHORT_REQ timer duration         */
   AP_UINT32       short_req_timeouts;   /* number of SHORT_REQ timeouts     */
   AP_UINT32       liveness_timeouts;    /* number of liveness timeouts      */
   AP_UINT32       in_invalid_sna_frames; /* number of invalid SNA frames    */
   AP_UINT32       in_sc_frames;         /* number of SC frames received     */
   AP_UINT32       out_sc_frames;        /* number of SC frames sent         */
   AP_INT32        delay_change_sum;     /* delay change sum                 */
   AP_UINT32       current_receiver_threshold;
                                         /* current ARB-R receiver threshold */
   AP_UINT32       minimum_receiver_threshold;
                                         /* minimum ARB-R receiver threshold */
   AP_UINT32       maximum_receiver_threshold;
                                         /* maximum ARB-R receiver threshold */
   AP_UINT32       sent_normals_count;   /* number of NORMALs sent           */
   AP_UINT32       sent_slowdowns_count; /* number of SLOWDOWNs sent         */
   AP_UINT32       rcvd_normals_count;   /* number of NORMALs received       */
   AP_UINT32       rcvd_slowdowns_count; /* number of SLOWDOWNs received     */
   AP_UINT32       dcs_reset_count_non_heal;
                                         /* number of non-healing resets     */
   AP_UINT16       dcs_reset_count_healing;
                                         /* number of self-healing resets    */
   unsigned char   arb_mode;             /* ARB mode (GREEN, YELLOW, RED)    */
   unsigned char   reserve[1];           /* reserved                         */
} RTP_STATISTICS;
```

提供されるパラメーター

提供されるパラメーター：

オペコード
AP_QUERY_RTP_CONNECTION

buf_ptr
リスト情報を書き込むことができるバッファーを指すポインター。 アプリケーションは VCB の終わりにデータを追加することができます。この場合、 *buf_ptr* は **ヌル**に設定されていなければなりません。

buf_size
提供されるバッファーのサイズ。

num_entries
RTP 接続の最大数 (データを戻す必要があります)。 特定の範囲ではなく、特定の接続のデータを要求するには、 1 という値を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定しま

す。この場合、 CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
各エントリーに必要な情報のレベル、および CS Linux がデータの戻りを開始するリスト内の位置。

以下のいずれかの値に必要な情報のレベルを指定します。

要約の要約
サマリー情報のみ。

追加の詳細
詳細情報。

以下のいずれかの値を指定して、論理 **それとも** 操作を使用してこの値を結合します。

リストの最初のリスト （ _R ）
リストの最初の項目から開始します。

リストを含む （ 包括的 ）
rtp_name パラメーターで指定されたエントリーから開始します。

次への AP_LIST_FROM_NEXT
rtp_name パラメーターによって指定されたエントリーの直後のエントリーから開始します。

rtp_name
RTP 接続の名前。 *list_options* パラメーターが **リストの最初のリスト** （ _R ）に設定されている場合、この値は無視されます。 これは 8 バイトの ASCII ストリングで、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than *buf_size*.

num_entries
The number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than *num_entries*.

rtp_connection_summary.overlay_size
The size of the returned `rtp_connection` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `rtp_connection_summary` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

rtp_connection_summary.rtp_name
Name of the RTP connection. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes.

rtp_connection_summary.first_hop_ls_name
Name of the link station of the first hop of the RTP connection. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes.

rtp_connection_summary.dest_node_name
Fully qualified name of the destination control point for the RTP portion of the session. The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

rtp_connection_summary.connection_type
Specifies the type of sessions on the RTP connection. Possible values are:

AP_RTP_CP_CP_SESSION
The RTP connection carries CP-CP sessions.

AP_RTP_LU_LU_SESSION
The RTP connection carries LU-LU sessions.

AP_RTP_ROUTE_SETUP
The RTP connection is used for route setup.

rtp_connection_summary.cos_name
Name of the class of service used by the RTP connection. This name is an EBCDIC string padded on the right with EBCDIC spaces.

rtp_connection_summary.num_sess_active
Number of sessions active on this RTP connection.

rtp_connection_detail.overlay_size
The size of the returned `rtp_connection` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `rtp_connection_detail` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

rtp_connection_detail.rtp_name
Name of the RTP connection. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes.

rtp_connection_detail.first_hop_ls_name
Name of the link station of the first hop of the RTP connection. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes.

rtp_connection_detail.dest_node_name
Fully qualified name of the destination control point for the RTP portion of the session. The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

rtp_connection_detail.isr_boundary_fn
Specifies whether the RTP Connection is being used for any ISR session for which the local node is providing HPR-APPN Boundary Function. Possible values are:

AP_YES
The RTP connection is being used for an ISR session for which the local node is providing HPR-APPN Boundary Function.

AP_NO
The RTP connection is not being used for an ISR session for which the local node is providing HPR-APPN Boundary Function.

rtp_connection_detail.connection_type
Specifies the type of sessions on the RTP connection. Possible values are:

AP_RTP_CP_CP_SESSION
The RTP connection carries CP-CP sessions.

AP_RTP_LU_LU_SESSION
The RTP connection carries LU-LU sessions.

AP_RTP_ROUTE_SETUP
The RTP connection is used for route setup.

rtp_connection_detail.cos_name
Name of the class of service used by the RTP connection. This name is an EBCDIC string padded on the right with EBCDIC spaces.

rtp_connection_detail.max_btu_size
Maximum size, in bytes, of the basic transmission unit (BTU) used on the RTP connection.

rtp_connection_detail.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.

rtp_connection_detail.local_tcid
Local TCID (transport control identifier) for the RTP connection.

rtp_connection_detail.remote_tcid
Remote TCID for the RTP connection.

rtp_connection_detail.rtp_stats.bytes_sent
Total number of bytes that the local node has sent on this RTP connection.

rtp_connection_detail.rtp_stats.bytes_received
Total number of bytes that the local node has received on this RTP connection.

rtp_connection_detail.rtp_stats.bytes_resent
Total number of bytes that the local node has resent on this RTP connection because bytes were lost in transit.

rtp_connection_detail.rtp_stats.bytes_discarded
Total number of bytes sent by the other end of the RTP connection that were discarded as duplicates of data already received.

rtp_connection_detail.rtp_stats.packets_sent
Total number of packets that the local node has sent on this RTP connection.

rtp_connection_detail.rtp_stats.packets_received
Total number of packets that the local node has received on this RTP connection.

rtp_connection_detail.rtp_stats.packets_resent
Total number of packets that the local node has resent on this RTP connection because packets were lost in transit.

rtp_connection_detail.rtp_stats.packets_discarded
Total number of packets sent by the other end of the RTP connection that were discarded as duplicates of data already received.

rtp_connection_detail.rtp_stats.gaps_detected
Total number of gaps detected by the local node. Each gap corresponds to one or more lost frames.

rtp_connection_detail.rtp_stats.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.

rtp_connection_detail.rtp_stats.max_send_rate
Maximum send rate on this RTP connection, measured in Kbits/second.

rtp_connection_detail.rtp_stats.min_send_rate
Minimum send rate on this RTP connection, measured in Kbits/second.

rtp_connection_detail.rtp_stats.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.

rtp_connection_detail.rtp_stats.max_receive_rate
Maximum receive rate on this RTP connection, measured in Kbits/second.

rtp_connection_detail.rtp_stats.min_receive_rate
Minimum receive rate on this RTP connection, measured in Kbits/second.

rtp_connection_detail.rtp_stats.burst_size
Current burst size on this RTP connection, measured in bytes.

rtp_connection_detail.rtp_stats.up_time
Total number of seconds this RTP connection has been active.

rtp_connection_detail.rtp_stats.smooth_rtt
Smoothed measure of round-trip time between the local node and the partner RTP node, measured in milliseconds.

rtp_connection_detail.rtp_stats.last_rtt
The last measured round-trip time between the local node and the partner RTP node, measured in milliseconds.

rtp_connection_detail.rtp_stats.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.

rtp_connection_detail.rtp_stats.short_req_timeouts
Total number of times the *short_req_ timer* has expired for this RTP connection.

rtp_connection_detail.rtp_stats.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.

rtp_connection_detail.rtp_stats.in_invalid_sna_frames
Total number of SNA frames received and discarded on this RTP connection because they were not valid.

rtp_connection_detail.rtp_stats.in_sc_frames
Total number of session control frames received on this RTP connection.

rtp_connection_detail.rtp_stats.out_sc_frames
Total number of session control frames sent on this RTP connection.

rtp_connection_detail.rtp_stats.delay_change_sum
Value of the delay change sum currently held by the ARB-R algorithm on this RTP connection.

rtp_connection_detail.rtp_stats.current_receiver_threshold
Value of the receiver threshold currently held by the ARB-R algorithm on this RTP connection.

rtp_connection_detail.rtp_stats.minimum_receiver_threshold
Value of the minimum receiver threshold currently held by the ARB-R algorithm on this RTP connection.

rtp_connection_detail.rtp_stats.maximum_receiver_threshold
Value of the maximum receiver threshold currently held by the ARB-R algorithm on this RTP connection.

rtp_connection_detail.rtp_stats.sent_normals_count
Number of NORMAL feedback ARB-R segments sent by the ARB-R algorithm on this RTP connection.

rtp_connection_detail.rtp_stats.sent_slowdowns_count
Number of SLOWDOWN1 and SLOWDOWN2 feedback ARB-R segments sent by the ARB-R algorithm on this RTP connection.

rtp_connection_detail.rtp_stats.rcvd_normals_count
Number of NORMAL feedback ARB-R segments received by the ARB-R algorithm on this RTP connection.

rtp_connection_detail.rtp_stats.rcvd_slowdowns_count
Number of SLOWDOWN1 and SLOWDOWN2 feedback ARB-R segments received by the ARB-R algorithm on this RTP connection.

rtp_connection_detail.rtp_stats.dcs_reset_count_non_heal
Number of delay change sum resets made as a part of normal ARB-R processing on this RTP connection.

rtp_connection_detail.rtp_stats.dcs_reset_count_healing
Number of delay change sum resets made to self-heal the ARB-R algorithm on this RTP connection.

rtp_connection_detail.rtp_stats.arb_mode
The current ARB-R status mode on this RTP connection. Possible values are:

0
GREEN

1
YELLOW

2
RED

rtp_connection_detail.num_sess_active
Number of sessions active on this RTP connection.

rtp_connection_detail.arb_mode
Specifies the ARB mode in use on this RTP Connection. Possible values are:

AP_ARB_S
Standard mode ARB.

AP_ARB_R
Responsive mode ARB.

AP_ARB_P
Progressive mode ARB.

AP_UNKNOWN
The ARB mode has not yet been determined because the RTP connection is not yet established.

rtp_connection_detail.refifo
Specifies whether refifo is enabled on the RTP connection. Possible values are:

AP_YES
Refifo is enabled. When CS Linux detects a gap in received data, it starts the refifo timer to allow time for out-of-sequence packets to arrive, and requests retransmission only if the packets are still missing when the timer expires.

AP_NO
Refifo is not enabled. When CS Linux detects a gap in received data, it requests retransmission of the missing packets immediately.

rtp_connection_detail.refifo_timer
The most recent refifo timer duration, in milliseconds.

rtp_connection_detail.path_switch_time
The time in seconds since the most recent path switch attempt on this RTP connection. If there have been no path switch attempts (*path_switch_atts* is set to zero), this parameter is set to zero.

rtp_connection_detail.path_switch_atts
The total number of path switch attempts made on this RTP connection.

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_RTP_CONNECTION
The *list_options* parameter was set to AP_LIST_INCLUSIVE to list all entries starting from the supplied name, but the *rtp_name* parameter was not valid.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

QUERY_RTP_TUNING

QUERY_RTP_TUNING returns information about the parameters that will be used for future RTP connections. This information was previously set up using DEFINE_RTP_TUNING.

VCB structure

```
typedef struct query_rtp_tuning
{
   AP_UINT16         opcode;                 /* Verb operation code             */
   unsigned char     reserv2;                /* reserved                        */
   unsigned char     format;                 /* reserved                        */
   AP_UINT16         primary_rc;             /* primary return code             */
   AP_UINT32         secondary_rc;           /* secondary return code           */
   unsigned char     path_switch_attempts;   /* number of path switch attempts  */
   unsigned char     short_req_retry_limit;  /* short request timer retry limit */
   AP_UINT16         path_switch_times[4];   /* path switch times               */
   AP_UINT32         refifo_cap;             /* maximum for refifo timer        */
   AP_UINT32         srt_cap;                /* maximum for short request timer */
   AP_UINT16         path_switch_delay;      /* minimum delay before path switch*/
   unsigned char     reserved[78];           /* reserved                        */
} QUERY_RTP_TUNING;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_RTP_TUNING

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

パス切り替えの試行回数
新規 RTP 接続での設定を試行するパス・スイッチの数。

不足 *req_retry_limit*
RTP 接続が切断され、パス・スイッチ処理を開始すると CS Linux が判断する前に、状況要求が送信される回数。

パス・スイッチタイム
CS Linux が切断された RTP 接続をパス・スイッチしようとする時間の長さ (秒)。 このパラメーターは、有効な伝送優先順位ごとに、 **低価格**、 **メディア・メディア**、 **上位** (**上位**)、および **ネットワーク**の 4 つの個別の時間制限として指定されます。

refifo_cap
RTP プロトコルは、 Re-FIFO Timer と呼ばれるタイマーを使用します。 このタイマーの値はプロトコルの一部として計算されますが、このパラメーターでは、タイマーを大きくできない最大値 (ミリ秒) が指定されます。 状況によっては、この最大値を設定すると、パフォーマンスが向上する場合 値 0 (ゼロ) は、タイマーが制限されておらず、プロトコルによって計算された値を取ることができることを意味します。

srt_cap
RTP プロトコルは、短時間要求タイマーと呼ばれるタイマーを使用します。 このタイマーの値はプロトコルの一部として計算されますが、このパラメーターでは、タイマーを大きくできない最大値 (ミリ秒) が指定されます。 状況によっては、この最大値を設定すると、パフォーマンスが向上する場合 値 0 (ゼロ) は、タイマーが制限されておらず、プロトコルによって計算された値を取ることができることを意味します。

パス・スイッチの遅延
パス・スイッチが発生するまでの最小遅延時間 (秒)。 遅延を指定すると、リモート・システムでの一時リソース不足による不要なパス・スイッチの試みが回避されます。特に、使用可能な経路が他にない場合です。

このパラメーターのデフォルト値はゼロです。これは、プロトコルが要求した場合にパス・スイッチの試行がすぐに発生する可能性があることを示しています。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

照会セキュリティー・アクセス・リスト

QUERY_SECURITY_ACCESS_LIST は、 CS Linux 構成ファイルに定義されているセキュリティー・アクセス・リストに関する情報を戻します。 使用されるオプションに応じて、単一のリストまたは複数のリストに関する情報を戻すことができます。

VCB 構造体

```
typedef struct query_security_access_list
{
  AP_UINT16         opcode;                 /* verb operation code           */
  unsigned char     reserv2;                /* reserved                      */
  unsigned char     format;                 /* reserved                      */
  AP_UINT16         primary_rc;             /* primary return code           */
  AP_UINT32         secondary_rc;           /* secondary return code         */
  unsigned char     *buf_ptr;               /* pointer to buffer             */
  AP_UINT32         buf_size;               /* buffer size                   */
  AP_UINT32         total_buf_size;         /* total buffer size required    */
  AP_UINT16         num_entries;            /* number of entries             */
  AP_UINT16         total_num_entries;      /* total number of entries       */
  unsigned char     list_options;           /* listing options               */
  unsigned char     reserv3;                /* reserved                      */
  unsigned char     list_name[14];          /* Security Access List name     */
  unsigned char     user_name[10];          /* user name                     */
  AP_UINT32         num_init_users;         /* number of users for first     */
                                            /* list when starting in middle  */
  AP_UINT32         num_last_users;         /* number of users on last       */
                                            /* overlay if last list is       */
                                            /* incomplete                    */
  unsigned char     last_list_incomplete;   /* set to AP_YES if user data    */
                                            /* for last list is incomplete   */
} QUERY_SECURITY_ACCESS_LIST;
```

```
typedef struct security_access_detail
{
  AP_UINT16         overlay_size;         /* size of returned entry          */
  unsigned char     list_name[14];        /* list name                       */
  unsigned char     reserv1[2];           /* reserved                        */
  AP_UINT32         num_filtered_users;   /* number of users returned        */
```

```
  SECURITY_LIST_DEF def_data;             /* list definition                 */
} SECURITY_ACCESS_DETAIL;
```

```
typedef struct security_list_def
{
  unsigned char     description[32];      /* description                     */
  unsigned char     reserv3[16];          /* reserved                        */
  AP_UINT32         num_users;            /* number of users in list         */
  unsigned char     reserv2[16];          /* reserved                        */
} SECURITY_LIST_DEF;
```

```
typedef struct security_user_data
{
  AP_UINT16         sub_overlay_size;     /* reserved                        */
  unsigned char     user_name[10];        /* user name                       */
} SECURITY_USER_DATA;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_SECURITY_ACCESS_LIST

buf_ptr
A pointer to a data buffer that CS Linux will use to return the requested information.

buf_size
Size of the supplied data buffer.

num_entries
Maximum number of security access lists for which data should be returned. 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).

To request data for a specific security access list rather than a range, specify the value 1. To return as many entries as possible, specify zero; in this case, CS Linux will return the maximum number of entries that can be accommodated in the supplied data buffer.

list_options
The position in the list from which CS Linux should begin to return data. Specify one of the following values:

AP_FIRST_IN_LIST
Start at the first user name for the first security access list.

AP_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.

AP_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.

The list is ordered by security access list name, and then by user name within each security access list. For more information about how the list is ordered and how the application can obtain specific entries from it, see “List options for QUERY_* Verbs” on page 34.

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 AP_FIRST_IN_LIST. The name is an ASCII string of 1-14 characters, padded on the right with spaces if the name is shorter than 14 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, set this parameter to 10 binary zeros.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

buf_size
バッファーに戻された情報の長さ。

total_buf_size
要求されたすべてのリスト情報を戻すために必要となるバッファーのサイズを示す戻り値。 これは、 *buf_size* より大きい場合があります

total_num_entries
戻された可能性がある項目の合計数。 これは、 *num_entries* より大きい場合があります

num_entries
実際に戻されたエントリーの数。 最後の項目が不完全である可能性があります。これは *last_list_* 不完全な パラメーターで示されます。

ユーザー数
ユーザー名 パラメーターがゼロ以外の値に設定されている場合、戻されたデータ内の最初のセキュリティー・アクセス・リストの情報がそのリストの最初のユーザーで開始されないようにすると、このパラメーターは、戻されたデータに含まれるこのリストのユーザー名構造の数を示します。 それ以外の場合、このパラメーターは使用されませ

***last_users* の数**
*last_list_* 不完全な パラメーターが、最後のリストのデータが不完全であることを示している場合、このパラメーターは、戻されたデータに含まれているこのリストのユーザー名構造の数を示します。 (このリストに対して返される フィルター・ファイル・ユーザー パラメーターは、使用可能なユーザー名構造の総数を示します。) それ以外の場合、このパラメーターは使用されませ

***last_list_*不完全な**
最後のセキュリティー・アクセス・リストの情報が不完全であるかどうかを示します 可能な値は次のとおりです

類人猿
最後のセキュリティー・アクセス・リストの完全なデータが、データ・バッファーに収めるには大きすぎます。 少なくとも 1 つのユーザー名構造が含まれていますが、データ・バッファーに組み込まれていないユーザー名構造がさらに存在します。 *last_users* の数 パラメーターは、戻されたユーザー名構造の数を示します。アプリケーションは、残りのデータを取得するためにさらに verb を発行することができます。

アプ・ノー
最後のリストのデータが完成します。

データ・バッファーの各項目は、次のもので構成されます。

セキュリティー・アクセスの詳細 *. list_name*
セキュリティー・アクセス・リストの名前。 これは 1 文字から 14 文字の ASCII ストリングです。

セキュリティー・アクセス詳細 *. num_filtered_users*
このセキュリティー・アクセス・リスト内のユーザー名の合計数。

security_access_detail.def_data.description
リストの定義に指定されている、セキュリティー・アクセス・リストを記述するヌルで終了するテキスト・ストリング。

セキュリティー・アクセス・ユーザー詳細 *. データ .num_users*
セキュリティー・アクセス・リスト内のユーザーの総数。

これがデータ・バッファーの最後のリストであり、 *last_list_*不完全な パラメーターが **類人猿**に設定されている場合、このリストに対して戻されるユーザー名構造の合計数は、 *last_users* の数 パラメーターで指定されたものになります。これは、 ユーザー数よりも小さくなります。

リスト内のユーザー名ごとに、 **セキュリティー・ ユーザー・ データ** 構造が以下の情報とともに戻されます。

ユーザー名
ユーザーの名前。

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

ファイル・ リスト名の変更
list_options パラメーターが **リストを含む** (**包括的**)に設定されましたが、 リスト名 パラメーターが、定義されたセキュリティー・アクセス・リストの名前と一致しませんでした。

ユーザー名の変更
list_options パラメーターが **リストを含む** (**包括的**)に設定されましたが、 ユーザー名 パラメーターが、指定されたセキュリティー・アクセス・リストに定義されているユーザー名と一致しませんでした。

ファイルの追加リスト・ オプション
list_options パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

QUERY_SESSION

QUERY_SESSION returns list information about sessions for a particular local LU.

This verb can be used to obtain either summary or detailed information, about a specific session or a range of sessions, depending on the options used.

This verb must be issued to a running node.

VCB structure

```
typedef struct query_session
{
   AP_UINT16       opcode;              /* verb operation code             */
   unsigned char   reserv2;             /* reserved                        */
   unsigned char   format;              /* reserved                        */
   AP_UINT16       primary_rc;          /* primary return code             */
   AP_UINT32       secondary_rc;        /* secondary return code           */
   unsigned char   *buf_ptr;            /* pointer to buffer               */
   AP_UINT32       buf_size;            /* buffer size                     */
   AP_UINT32       total_buf_size;      /* total buffer size required      */
   AP_UINT16       num_entries;         /* number of entries               */
   AP_UINT16       total_num_entries;   /* total number of entries         */
   unsigned char   list_options;        /* listing options                 */
   unsigned char   reserv3;             /* reserved                        */
   unsigned char   lu_name[8];          /* LU name                         */
   unsigned char   lu_alias[8];         /* LU alias                        */
   unsigned char   plu_alias[8];        /* partner LU alias                */
```

```
   unsigned char   fqplu_name[17];       /* fully qualified partner LU name  */
   unsigned char   mode_name[8];         /* mode name                        */
   unsigned char   session_id[8];        /* session ID                       */
} QUERY_SESSION;
```

```
typedef struct session_summary
{
   AP_UINT16       overlay_size;         /* size of returned entry           */
   unsigned char   plu_alias[8];         /* partner LU alias                 */
   unsigned char   fqplu_name[17];       /* fully qualified partner LU name  */
   unsigned char   reserv3[1];           /* reserved                         */
   unsigned char   mode_name[8];         /* mode name                        */
   unsigned char   session_id[8];        /* session ID                       */
   FQPCID          fqpcid;               /* fully qualified procedure        */
                                         /*  correlator ID                   */
} SESSION_SUMMARY;
```

```
typedef struct session_detail
{
   AP_UINT16       overlay_size;         /* size of returned entry           */
   unsigned char   plu_alias[8];         /* partner LU alias                 */
   unsigned char   fqplu_name[17];       /* fully qualified partner LU name  */
   unsigned char   reserv3[1];           /* reserved                         */
   unsigned char   mode_name[8];         /* mode name                        */
   unsigned char   session_id[8];        /* session ID                       */
   FQPCID          fqpcid;               /* fully qualified procedure        */
                                         /*  correlator ID                   */
   unsigned char   cos_name[8];          /* Class of Service name            */
   unsigned char   trans_pri;            /* Transmission priority:           */
   unsigned char   ltd_res;              /* Session spans a limited resource */
   unsigned char   polarity;             /* Session polarity                 */
   unsigned char   contention;           /* Session contention               */
   SESSION_STATS   sess_stats;           /* Session statistics               */
   unsigned char   reserv3a;             /* reserved                         */
   unsigned char   sscp_id[6];           /* SSCP ID of host                  */
   unsigned char   reserva;              /* reserved                         */
   AP_UINT32       session_start_time;   /* start time of the session        */
   AP_UINT16       session_timeout;      /* session timeout                  */
   unsigned char   cryptography;         /* reserved                         */
   unsigned char   reservb[5];           /* reserved                         */
   unsigned char   comp_in_series;       /* reserved                         */
   unsigned char   plu_slu_comp_lvl;     /* PLU to SLU compression level     */
   unsigned char   slu_plu_comp_lvl;     /* SLU to PLU compression level     */
   unsigned char   rscv_len;             /* Length of following RSCV         */
} SESSION_DETAIL;
```

The session detail structure may be followed by a 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.

```
typedef struct fqpcid
{
   unsigned char   pcid[8];              /* procedure correlator identifier  */
   unsigned char   fqcp_name[17];        /* originator's network qualified   */
                                         /*  CP name                         */
   unsigned char   reserve3[3];          /* reserved                         */
} FQPCID;
```

```
typedef struct session_stats
{
   AP_UINT16       rcv_ru_size;          /* session receive RU size          */
   AP_UINT16       send_ru_size;         /* session send RU size             */
   AP_UINT16       max_send_btu_size;    /* Maximum send BTU size            */
   AP_UINT16       max_rcv_btu_size;     /* Maximum rcv BTU size             */
   AP_UINT16       max_send_pac_win;     /* Maximum send pacing window size  */
   AP_UINT16       cur_send_pac_win;     /* Current send pacing window size  */
   AP_UINT16       max_rcv_pac_win;      /* Maximum receive pacing window    */
                                         /*  size                            */
   AP_UINT16       cur_rcv_pac_win;      /* Current receive pacing window    */
                                         /*  size                            */
   AP_UINT32       send_data_frames;     /* Number of data frames sent       */
   AP_UINT32       send_fmd_data_frames; /* Num fmd data frames sent         */
   AP_UINT32       send_data_bytes;      /* Number of data bytes sent        */
   AP_UINT32       rcv_data_frames;      /* Number of data frames received   */
   AP_UINT32       rcv_fmd_data_frames;  /* Num fmd data frames received     */
```

```
    AP_UINT32       rcv_data_bytes;       /* Number of data bytes received    */
    unsigned char   sidh;                 /* Session ID high byte (from LFSID)*/
    unsigned char   sidl;                 /* Session ID low byte (from LFSID) */
    unsigned char   odai;                 /* ODAI bit set                     */
    unsigned char   ls_name[8];           /* Link station name (or RTP name)  */
    unsigned char   pacing_type;          /* type of pacing in use            */
} SESSION_STATS;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
AP_QUERY_SESSION

buf_ptr
CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

num_entries
データが戻されるセッションの最大数。 特定の範囲ではなく、特定のセッションのデータを要求するには、 1 という値を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、 CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータの戻りを開始する必要のあるリスト内の位置、および各項目に必要な情報のレベル。 以下の値のいずれかを使用して、情報のレベルを指定します。

要約の要約
サマリー情報のみ。

追加の詳細
詳細情報。

以下のいずれかの値を指定して、論理 **それとも** 操作を使用してこの値を結合します。

リストの最初のリスト （ _R ）
リストの最初の項目から開始します。

リストを含む （ 包括的 ）
セッション *ID* パラメーターで指定されたエントリーから開始します。

次への AP_LIST_FROM_NEXT
セッション *ID* パラメーターによって指定されたエントリーの直後のエントリーから開始します。

list_options パラメーターが **リストを含む** （ **包括的** ）または **次への** AP_LIST_FROM_NEXT に設定されている場合は、ローカル LU (*lu_name* または *lu_alias*)、パートナー LU (*plu_alias* または *fqplu_name*)、および指定された モード名 の組み合わせがセッションのリストへの索引として使用されます。

リストの順序とアプリケーションでの特定のエントリーの取得方法について詳しくは、 34 ページの『List options for QUERY_* Verbs』を参照してください。

lu_name
LU 名。 これは 8 バイトのタイプ A の EBCDIC ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。 LU が LU 名ではなくその別名によって識別されることを指定するには、このパラメーターを 8 桁の 2 進ゼロに設定し、次のパラメーターに LU 別名を指定します。 ローカル CP (デフォルトの LU) に関連した LU を指定するには、 *lu_name* と *lu_alias* の両方を 2 進ゼロに設定します。

lu_alias
ローカルに定義された LU の別名。 これは 8 バイトの ASCII ストリングで、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。 このパラメーターが使用されるのは、 *lu_name* が 8 進ゼロに設定されている場合のみです。それ以外の場合は ローカル CP (デフォルトの LU) に関連した LU を指定するには、 *lu_name* と *lu_alias* の両方を 2 進ゼロに設定します。

plu_alias
パートナー LU の別名。 特定のパートナー LU に関連付けられたセッションに関する情報のみを戻すには、パートナー LU の別名 (このパラメーターに) またはパートナー LU の完全修飾名を指定します (以下のパラメーターを参照) 。 パートナー LU でフィルター操作しないですべてのセッションに関する情報を戻すには、これらのパラメーターの両方を 2 進ゼロに設定します。

これは 8 バイトの ASCII ストリングで、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。 LU がその別名ではなくその LU 名によって識別されるように指定するには、このパラメーターを 8 進ゼロに設定し、次のパラメーターに LU 名を指定します。

fqplu_name
パートナー LU の完全修飾ネットワーク名。 このパラメーターが使用されるのは、 *plu_alias* が 8 進ゼロに設定されている場合のみです。それ以外の場合は

この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

モード名
モード名フィルター。 特定のモードに関連付けられているセッションに関する情報のみを戻すには、モード名を指定します。また、パートナー LU も指定する必要があります (上記の 2 つのパラメーターのいずれかを使用します) 。 モード名をフィルタリングせずに、すべてのセッションに関する情報を戻すには、このパラメーターを 8 進ゼロに設定します。

モード名は 8 バイトのタイプ A の EBCDIC ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

セッション *ID*
8 バイトのセッション ID 。 *list_options* を **リストの最初のリスト** (_R)に設定すると、このパラメーターは無視されます。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

session_summary.overlay_size
The size of the returned `session_summary` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `session_summary` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

session_summary.plu_alias
Partner LU alias. This is an 8-byte ASCII character string, right-padded with ASCII spaces.

session_summary.fqplu_name
Fully qualified network name for the partner LU. The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

session_summary.mode_name
Mode name. This is an 8-byte type-A EBCDIC string (starting with a letter), right-padded with EBCDIC spaces.

session_summary.session_id
8-byte identifier of the session.

session_summary.fqpcid.pcid
Procedure Correlator ID. This is an 8-byte hexadecimal string.

session_summary.fqpcid.fqcp_name
Fully qualified CP name. The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

session_detail.overlay_size
The size of the returned `session_detail` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `session_detail` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

session_detail.plu_alias
Partner LU alias. This is an 8-byte ASCII character string, right-padded with ASCII spaces.

session_detail.fqplu_name
Fully qualified network name for the partner LU. The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

session_detail.mode_name
Mode name. This is an 8-byte type-A EBCDIC string (starting with a letter), right-padded with EBCDIC spaces.

session_detail.session_id
8-byte identifier of the session.

session_detail.fqpcid.pcid
Procedure Correlator ID. This is an 8-byte hexadecimal string.

session_detail.fqpcid.fqcp_name
Fully qualified control point name. This is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of 1-8 A-string characters, an EBCDIC dot (period) character, and a network name of 1-8 A-string characters.

session_detail.cos_name
Class of service name. This is an 8-byte type-A EBCDIC string, padded on the right with spaces if the name is shorter than 8 characters.

session_detail.trans_pri
Transmission priority. Possible values are:

`AP_LOW`

`AP_MEDIUM`

`AP_HIGH`

AP_NETWORK

session_detail.ltd_res
Specifies whether the session uses a limited resource link. Possible values are:

AP_YES
Session uses a limited resource link.

AP_NO
Session does not use a limited resource link.

session_detail.polarity
Specifies the polarity of the session. Possible values are:

AP_PRIMARY

AP_SECONDARY

session_detail.contention
Specifies whether the session is a contention winner or contention loser session for the local LU. Possible values are:

AP_CONWINNER
Contention winner session

AP_CONLOSER
Contention loser session

session_detail.sess_stats.rcv_ru_size
Maximum receive RU size.

session_detail.sess_stats.send_ru_size
Maximum send RU size.

session_detail.sess_stats.max_send_btu_size
Maximum BTU size that can be sent.

session_detail.sess_stats.max_rcv_btu_size
Maximum BTU size that can be received.

session_detail.sess_stats.max_send_pac_win
Maximum size of the send pacing window on this session.

session_detail.sess_stats.cur_send_pac_win
Current size of the send pacing window on this session.

session_detail.sess_stats.max_rcv_pac_win
Maximum size of the receive pacing window on this session.

session_detail.sess_stats.cur_rcv_pac_win
Current size of the receive pacing window on this session.

session_detail.sess_stats.send_data_frames
Number of normal flow data frames sent.

session_detail.sess_stats.send_fmd_data_frames
Number of normal flow FMD data frames sent.

session_detail.sess_stats.send_data_bytes
Number of normal flow data bytes sent.

session_detail.sess_stats.rcv_data_frames
Number of normal flow data frames received.

session_detail.sess_stats.rcv_fmd_data_frames
Number of normal flow FMD data frames received.

session_detail.sess_stats.rcv_data_bytes
Number of normal flow data bytes received.

session_detail.sess_stats.sidh
Session ID high byte.

session_detail.sess_stats.sidl
Session ID low byte.

session_detail.sess_stats.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. It sets it to one if the BIND sender is the node containing the secondary link station.

session_detail.sess_stats.ls_name
Link station name associated with statistics. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes. This field can be used to correlate the session statistics with the link over which session data flows.

session_detail.sess_stats.pacing_type
The type of receive pacing in use on this session. Possible values are:

- AP_NONE
- AP_FIXED
- AP_ADAPTIVE

session_detail.duplex_support
Returns the conversation duplex support as negotiated on the BIND. Possible values are:

AP_HALF-DUPLEX
Only half-duplex conversations are supported.

AP_FULL_DUPLEX
Both full-duplex and half-duplex sessions are supported. Expedited data is also supported.

session_detail.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_detail.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_detail.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)

session_detail.plu_slu_comp_lvl
Specifies the compression level for data sent from the primary LU (PLU) to the secondary LU (SLU). Possible values are:

AP_NONE
Compression is not used.

AP_RLE_COMPRESSION
Run-length encoding (RLE) compression is used.

AP_LZ9_COMPRESSION
LZ9 compression is used.

AP_LZ10_COMPRESSION
LZ10 compression is used.

session_detail.slu_plu_comp_lvl
Specifies the compression level for data sent from the secondary LU (SLU) to the primary LU (PLU). Possible values are:

AP_NONE
Compression is not used.

AP_RLE_COMPRESSION
Run-length encoding (RLE) compression is used.

AP_LZ9_COMPRESSION
LZ9 compression is used.

AP_LZ10_COMPRESSION
LZ10 compression is used.

session_detail.rscv_len
Length of the RSCV which is appended to the `session_detail` structure. (If none is appended, then the length is zero.)

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_LU_ALIAS
The specified *lu_alias* parameter was not valid.

AP_INVALID_LU_NAME
The specified *lu_name* parameter was not valid.

AP_INVALID_SESSION_ID
The *list_options* parameter was set to AP_LIST_INCLUSIVE to list all entries starting from the supplied value, but the *session_id* parameter was not valid.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

照会ネット・ネット

QUERY_SNA_NET は、 **スネネット** ファイルで定義されているように、バックアップ・サーバーとして機能できるサーバーに関する情報を戻します。 これは、使用されるオプションに応じて、特定のサーバーまたは複数のサーバーに関する情報を取得するために使用できます。

このファイル内のサーバー名の順序は重要です。ファイルにリストされている最初のサーバーは、アクティブである場合は常にコントローラーになり、 2 番目のサーバーは、最初のサーバーが非アクティブの場合はコントローラーになり、 3 番目のサーバーはコントローラーになり、 2 番目のサーバーが非アクティブの場合はコントローラーになり、 2 番目のサーバーはコントローラーになります。 このため、 QUERY_SNA_NET で戻されたサーバー名のリストは、ファイルと同じ順序になっています。戻された名前は、他の QUERY_* verb と同様に、名前の長さと辞書式順序で配列されることはありません。

This verb must be issued to the **スネネット** ファイル。

VCB structure

```
typedef struct query_sna_net
{
```

```
   AP_UINT16        opcode;                 /* Verb operation code          */
   unsigned char    reserv2;                /* reserved                     */
   unsigned char    format;                 /* reserved                     */
   AP_UINT16        primary_rc;             /* Primary return code          */
   AP_UINT32        secondary_rc;           /* Secondary return code        */
   unsigned char    *buf_ptr;               /* pointer to buffer            */
   AP_UINT32        buf_size;               /* buffer size                  */
   AP_UINT32        total_buf_size;         /* total buffer size required   */
   AP_UINT16        num_entries;            /* number of entries            */
   AP_UINT16        total_num_entries;      /* total number of entries      */
   unsigned char    list_options;           /* listing options              */
   unsigned char    security;               /* reserved                     */
   unsigned char    domain_name[64];        /* domain name                  */
   unsigned char    server_name[128];    /* controller or backup server name */
   unsigned char    reserv4[4];             /* reserved                     */
} QUERY_SNA_NET;
```

```
typedef struct backup_summary
{
   AP_UINT16        overlay_size;           /* size of returned entry       */
   unsigned char    reserv1[2];             /* reserved                     */
   unsigned char    server_name[128];    /* controller or backup server name */
   unsigned char    reserv2[4];             /* reserved                     */
} BACKUP_SUMMARY;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_SNA_NET

buf_ptr
A pointer to a data buffer that CS Linux will use to return the requested information.

buf_size
Size of the supplied data buffer.

num_entries
Maximum number of server names for which data should be returned. To request a specific entry rather than a range, specify the value 1. To return as many entries as possible, specify zero; in this case, CS Linux will return the maximum number of entries that can be accommodated in the supplied data buffer.

list_options
The position in the list from which CS Linux should begin to return data.

Specify one of the following values:

AP_FIRST_IN_LIST
Start at the first entry in the list.

AP_LIST_INCLUSIVE
Start at the entry specified by the *server_name* parameter.

AP_LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *server_name* parameter.

For more information about how the application can obtain specific entries from the list, see “List options for QUERY_* Verbs” on page 34. The server names are listed in the same order as in the file, not in order of name length and/or lexicographical order as for other QUERY_* verbs.

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 AP_FIRST_IN_LIST.

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

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

domain_name
The name of the TCP/IP domain containing the CS Linux LAN. This name was specified during installation of the controller server.

Each entry in the data buffer consists of the following parameters:

backup_summary.overlay_size
The size of the returned `backup_summary` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `backup_summary` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

backup_summary.server_name
Server name.

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc

ファイルの追加リスト・オプション
list_options パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: state check

If the verb does not execute because of a state check, CS Linux returns the following parameters.

primary_rc
AP_STATE_CHECK

secondary_rc
Possible values are:

AP_RECORD_NOT_FOUND
The *list_options* parameter was set to AP_LIST_INCLUSIVE or AP_LIST_FROM_NEXT to list entries starting from the supplied server name, but the *backup_name* parameter did not match an entry in the file. If the supplied name was one returned on a previous QUERY_SNA_NET verb, this indicates that the list has been updated (by another administration program or NOF application) since the previous verb; the application should reissue QUERY_SNA_NET to obtain the complete list.

AP_INVALID_TARGET
The target handle on the NOF API call specified a configuration file or a node. This verb must be issued to the `sna.net` file.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_STATE_CHECK, which are common to all NOF verbs.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

QUERY_STATISTICS

QUERY_STATISTICS returns statistics on the usage of an LS or port. The MPC link type does not support link statistics; do not issue this verb for an MPC LS or port. The QLLC link type does not support link statistics; do not issue this verb for a QLLC LS or port.

The type of information returned depends on the DLC type:

For SDLC, the verb 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 Ringor Ethernet, the verb returns statistics information for either an LS or a port.

For Enterprise Extender, the verb returns statistics information for an LS.

This verb must be issued to a running node.

VCB 構造体

```
typedef struct query_statistics
{
   AP_UINT16        opcode;                 /* verb operation code          */
   unsigned char    reserv2;                /* reserved                     */
   unsigned char    format;                 /* reserved                     */
   AP_UINT16        primary_rc;             /* primary return code          */
   AP_UINT32        secondary_rc;           /* secondary return code        */
   unsigned char    name[8];                /* LS name or port name         */
   unsigned char    stats_type;             /* LS or port statistics?       */
   unsigned char    table_type;             /* statistics table requested   */
   unsigned char    reset_stats;            /* reset the statistics?        */
   unsigned char    dlc_type;               /* type of DLC                  */
   unsigned char    statistics[256];        /* current statistics           */
   unsigned char    reserva[20];            /* reserved                     */
} QUERY_STATISTICS;
```

SDLC の LS 統計 :

```
typedef struct sdl_ls_stats_table
{
  V0_MUX_INFO    mux_info;          /* streams config info                  */
  AP_UINT32      index;             /* index of port that owns LS           */
  unsigned int   address;           /* poll address of secondary link station*/
  unsigned char  reserv[3];         /* reserved                             */
  AP_UINT32      blus_in;           /* frames received from adjacent link   */
                                    /*  station                             */
  AP_UINT32      blus_out;          /* frames sent to adjacent link station */
  AP_UINT32      octets_in;         /* bytes received from adjacent link    */
```

```
                                  /*  station                              */
  AP_UINT32      octets_out;      /* bytes sent to adjacent link station   */
  AP_UINT32      polls_out;       /* polls sent to adjacent link station   */
  AP_UINT32      poll_rsps_out;   /* polls responded to by adjacent link   */
                                  /*  station                              */
  AP_UINT32      local_busies;    /* number of times local link station has*/
                                  /*  entered busy state (RNR)            */
  AP_UINT32      remote_busies;   /* number of times remote link station   */
                                  /*  has entered busy state (RNR)        */
  AP_UINT32      iframes_in;      /* I-frames rcvd from adjacent link      */
                                  /*  station                             */
  AP_UINT32      iframes_out;     /* I-frames sent to adjacent link station*/
  AP_UINT32      retransmits_in;  /* Total number of retransmitted         */
                                  /* I-frames received
  AP_UINT32      retransmits_out; /* I-frames retransmitted since LS       */
                                  /*  start-up                            */
  AP_UINT32      ioctets_in;      /* bytes in I-frames received            */
  AP_UINT32      ioctets_out;     /* bytes in I-frames sent                */
  AP_UINT32      uiframes_in;     /* reserved                              */
  AP_UINT32      uiframes_out;    /* reserved                              */
  AP_UINT32      xids_in;         /* XIDs rcvd from adjacent link station  */
  AP_UINT32      xids_out;        /* XIDs sent to adjacent link station    */
  AP_UINT32      tests_in;        /* TEST frames received                  */
  AP_UINT32      tests_out;       /* TEST frames sent                      */
  AP_UINT32      rejs_in;         /* REJ frames received                   */
  AP_UINT32      rejs_out;        /* REJ frames sent                       */
  AP_UINT32      frmrs_in;        /* FRMR frames received                  */
  AP_UINT32      frmrs_out;       /* FRMR frames sent                      */
  AP_UINT32      sims_in;         /* SIM frames received                   */
  AP_UINT32      sims_out;        /* SIM frames sent                       */
  AP_UINT32      rims_in;         /* RIM frames received                   */
  AP_UINT32      rims_out;        /* RIM frames sent                       */
  AP_UINT32      disc_in;         /* reserved                              */
  AP_UINT32      disc_out;        /* reserved                              */
  AP_UINT32      ua_in;           /* reserved                              */
  AP_UINT32      ua_out;          /* reserved                              */
  AP_UINT32      dm_in;           /* reserved                              */
  AP_UINT32      dm_out;          /* reserved                              */
  AP_UINT32      snrm_in;         /* SNRM frames received                  */
  AP_UINT32      snrm_out;        /* SNRM frames sent                      */
} SDL_LS_STATS_TABLE;
```

SDLC の LS 操作情報 :

```
typedef struct sdl_ls_oper_table
{
  V0_MUX_INFO    mux_info;       /* streams config info                   */
  AP_UINT32      index;          /* index of port that owns LS            */
  unsigned char  address;        /* poll address of secondary link station*/
  unsigned char  reserve;        /* reserved                              */
  AP_UINT16      role;           /* current role of link station          */
  unsigned char  name[8];        /* reserved                              */
  AP_UINT16      state;          /* operational state of LS               */
  AP_UINT16      maxdata;        /* current max PDU size for logical link */
  AP_UINT32      replyto;        /* current reply timeout                 */
  AP_UINT32      maxin;          /* current max unack'd frames LS can receive*/
  AP_UINT32      maxout;         /* current max unack'd frames LS can send*/
  unsigned char  modulo;         /* sequence number modulus               */
  unsigned char  reserv2[3];     /* reserved                              */
  AP_UINT32      retries_m;      /* number of retries in a retry sequence */
  AP_UINT32      retries_t;      /* interval between retry sequences      */
  AP_UINT32      retries_n;      /* number of times to repeat retry sequence */
  AP_UINT32      rnrlimit;       /* how long adjacent LS can be in RNR state */
                                 /*  before it is considered inoperative  */
  unsigned char  datmode;        /* communications mode with adjacent LS  */
  unsigned char  last_fail_cause;         /* reserved                     */
  unsigned char  last_fail_ctrl_in[2];    /* control field of last frame rcvd*/
                                 /*  before last failure                  */
  unsigned char  last_fail_ctrl_out[2];   /* control field of last frame sent*/
                                 /*  before last failure                  */
  unsigned char  last_fail_frmr_info[5];  /* info field of FRMR frame if   */
                                 /*  last failure was caused by           */
                                 /*  invalid frame                        */
  unsigned char  sdoppad1;               /* reserved                      */
  AP_UINT32      last_fail_replyto_s;    /* number of REPLYTO timeouts at */
                                 /*  time of last failure                 */
  unsigned char  g_poll;         /* group poll address                    */
  unsigned char  sim_rim;        /* are SIM / RIM supported?              */
  unsigned char  xmit_rcv_cap;   /* transmit / receive capability         */
} SDL_LS_OPER_TABLE;
```

SDLC のポート統計 :

```
typedef struct sdl_port_stats_table
{
  V0_MUX_INFO    mux_info;        /* streams config info                    */
  AP_UINT32      index;           /* index of port                          */
  AP_UINT32      dwarf_frames;    /* frames received too short to be valid  */
  AP_UINT32      polls_out;       /* polls sent to adjacent link stations   */
  AP_UINT32      poll_rsps_out;   /* polls responded to by adjacent link stns*/
  AP_UINT32      local_busies;    /* number of times local link station     */
                                  /*  has entered busy state (RNR)          */
  AP_UINT32      remote_busies;   /* number of times remote link stations   */
                                  /*  have entered busy state (RNR)         */
  AP_UINT32      iframes_in;      /* I-frames rcvd from adjacent link       */
                                  /*  stations                              */
  AP_UINT32      iframes_out;     /* I-frames sent to adjacent link stations */
  AP_UINT32      octets_in;       /* bytes received from adjacent link      */
                                  /*  stations                              */
  AP_UINT32      octets_out;      /* bytes sent to adjacent link stations   */
  AP_UINT32      protocol_errs;   /* link deactivations due to bad rcvd     */
                                  /*  frames                                */
  AP_UINT32      activity_to_s;   /* link deactivations due to inactivity   */
  AP_UINT32      rnrlimit_s;      /* link deacts due to rem busy timer expiry*/
  AP_UINT32      retries_exps;    /* link deacts due to end of retry sequence*/
  AP_UINT32      retransmits_in;  /* retransmitted I-frames rcvd since      */
                                  /*  start-up                              */
  AP_UINT32      retransmits_out; /* I-frames retransmitted  since start-up  */
} SDL_PORT_STATS_TABLE;
```

SDLC のポート操作情報 :

```
typedef struct sdl_port_oper_table
{
  V0_MUX_INFO     mux_info;            /* streams config info               */
  AP_UINT32       index;               /* index of port                     */
  unsigned char   name[8];             /* reserved                          */
  unsigned char   role;                /* current role of link station(s)   */
                                       /*  using port                       */
  unsigned char   type;                /* line type - leased or switched    */
  unsigned char   topology;            /* can port be point-to-point or     */
                                       /*  multipoint                       */
  unsigned char   reserve;             /* reserved                          */
  AP_UINT32       activto;             /* how long switched line can be     */
                                       /*  inactive before port disconnects  */
  AP_UINT32       pause;               /* time between poll cycles          */
  unsigned char   slow_poll_method;    /* slow poll method                  */
  unsigned char   reserv2[3];          /* reserved                          */
  AP_UINT32       slow_poll_timer;     /* slow poll timer                   */
  unsigned char   last_fail_cause;     /* reserved                          */
} SDL_PORT_OPER_TABLE;
```

トークンリング、イーサネットの LS 統計

```
typedef struct llc2_ls_stats
{
   V0_MUX_INFO      mux_info;             /* streams config info           */
   unsigned char    local_mac[6];         /* MAC address of local port     */
   unsigned char    local_sap;            /* SAP number of local port      */
   unsigned char    reserve1;             /* reserved                      */
   unsigned char    remote_mac[6];        /* MAC address of remote port    */
   unsigned char    remote_sap;           /* SAP number of remote port     */
   unsigned char    reserve2;             /* reserved                      */
   AP_UINT16        rif_len;              /* length of RIF data for TR     */
   AP_UINT16        rif[8];               /* Routing Information Field     */
                                          /*  for TR                       */
   unsigned char    ls_fsm;               /* LLC2 FSM state                */
   unsigned char    reserve3;             /* reserved                      */
   AP_UINT16        mac_type;             /* LS MAC type                   */
   AP_UINT16        max_btu_size;         /* maximum BTU size              */
   AP_UINT16        send_window;          /* send window                   */
   AP_UINT16        receive_window;       /* receive window                */
   AP_UINT32        t1_expiry_count;      /* T1 expiry count               */
   AP_UINT32        t2_expiry_count;      /* T2 expiry count               */
   AP_UINT32        remote_busy;          /* remote busy state count       */
   AP_UINT32        local_busy;           /* local busy state count        */
   AP_UINT32        i_frames_sent;        /* count of I-frames sent        */
   AP_UINT32        i_bytes_sent;         /* count of bytes in I-frames sent*/
   AP_UINT32        i_frames_rcvd;        /* count of I-frames received    */
```

```
   AP_UINT32        i_bytes_rcvd;          /* count of bytes in I-frames     */
                                           /*  received                      */
   AP_UINT32        i_frames_rjctd;        /* count of I-frames rejected     */
   AP_UINT32        i_bytes_rjctd;         /* count of bytes in I-frames     */
                                           /*  rejected                      */
   AP_UINT32        i_frames_rexmit;       /* count of I-frames retransmitted*/
   AP_UINT32        i_bytes_rexmit;        /* count of bytes in I-frms       */
                                           /*  rexmitted                     */
   AP_UINT32        rej_frames_sent;       /* count of REJ frames sent       */
   AP_UINT32        rej_frames_rcvd;       /* count of REJ frames received   */
   AP_UINT32        xid_frames_sent;       /* count of XID frames sent       */
   AP_UINT32        xid_frames_rcvd;       /* count of XID frames received   */
   AP_UINT16        ack_timeout;           /* acknowledgment timeout         */
   AP_UINT16        p_bit_timeout;         /* poll bit timeout               */
   AP_UINT16        t2_timeout;            /* T2 timeout                     */
   AP_UINT16        rej_timeout;           /* REJ timeout                    */
   AP_UINT16        busy_state_timeout;    /* busy state timeout             */
   AP_UINT16        idle_timeout;          /* idle timeout                   */
   AP_UINT16        max_retry;             /* max retry count                */
} LLC2_LS_STATS;
```

トークンリング、イーサネットのポート統計 :

```
typedef struct llc2_port_stats
{
   V0_MUX_INFO      mux_info;              /* streams config info            */
   AP_UINT32        time_secs;             /* system time when port was      */
                                           /*  activated                     */
   AP_UINT16        time_ms;               /* system time when port was      */
                                           /*  activated                     */
   unsigned char    mac_addr[6];           /* MAC address of port            */
   AP_UINT16        ack_timeout;           /* reserved                       */
   AP_UINT16        p_bit_timeout;         /* reserved                       */
   AP_UINT16        t2_timeout;            /* reserved                       */
   AP_UINT16        rej_timeout;           /* reserved                       */
   AP_UINT16        busy_state_timeout;    /* reserved                       */
   AP_UINT16        idle_timeout;          /* reserved                       */
   AP_UINT16        max_retry;             /* reserved                       */
   AP_UINT16        max_btu_size;          /* max BTU size for port          */
   AP_UINT16        ls_count;              /* count of LSs using port        */
   AP_UINT16        reserve1;              /* reserved                       */
   AP_UINT32        ui_frames_sent;        /* count of UI frames sent        */
   AP_UINT32        ui_frames_rcvd;        /* count of UI frames received    */
   LLC2_DEV_STATS   device_stats;          /* device statistics              */
} LLC2_PORT_STATS;
```

```
typedef struct llc2_dev_stats
{
   unsigned char    adapter_number;        /* reserved                       */
   unsigned char    res1;                  /* reserved                       */
   unsigned char    line_error;            /* line error count               */
   unsigned char    internal_error;        /* internal error count           */
   unsigned char    burst_error;           /* burst error count              */
   unsigned char    ari_fci_error;         /* ARI/FCI error count            */
   unsigned char    end_delim;             /* end delimiter                  */
   unsigned char    res2;                  /* reserved                       */
   unsigned char    lost_frame;            /* lost frame count               */
   unsigned char    rcv_cngstn;            /* Receive congestion count       */
   unsigned char    frm_cpy_err;           /* Frame Copied error count       */
   unsigned char    freq_err;              /* frequency error count          */
   unsigned char    token_err;             /* token error count              */
   unsigned char    crc_err;               /* CRC error count                */
   unsigned char    res3;                  /* reserved                       */
   unsigned char    xmit_err;              /* transmit error count           */
   unsigned char    res4;                  /* reserved                       */
   unsigned char    collision_err;         /* collision error count          */
   unsigned char    res5[7];               /* reserved                       */
} LLC2_DEV_STATS;
```

エンタープライズ・エクステンダーの LS 統計 :

```
typedef struct udp_ls_stats_table
{
   V0_MUX_INFO      mux_info;           /* streams config info               */
   AP_UINT32        udp_low_out;        /* Count of UDP datagrams sent       */
                                        /* containing low priority data      */
   AP_UINT32        udp_med_out;        /* Count of UDP datagrams sent       */
                                        /* containing medium priority data   */
```

```
   AP_UINT32        udp_high_out;       /* Count of UDP datagrams sent       */
                                        /* containing high priority data     */
   AP_UINT32        udp_network_out;    /* Count of UDP datagrams sent       */
                                        /* containing network priority data  */
   AP_UINT32        udp_llc_out;        /* Count of UDP datagrams sent       */
                                        /* containing LLC commands           */
} UDP_LS_STATS_TABLE;
```

```
typedef struct v0_mux_info
{
   AP_UINT16         dlc_type;               /* DLC implementation type     */
   unsigned char     need_vrfy_fixup;        /* reserved                    */
   unsigned char     num_mux_ids;            /* reserved                    */
   AP_UINT32         card_type;              /* type of adapter card        */
   AP_UINT32         adapter_number;         /* DLC adapter number          */
   AP_UINT32         oem_data_length;        /* reserved                    */
   AP_INT32          mux_ids[5];             /* reserved                    */
} V0_MUX_INFO;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_STATISTICS

name
Name of the LS or port for which statistics are required (as specified by the *stats_type* parameter). This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes. CS Linux uses this name to correlate the response to the correct link station or port.

stats_type
The type of resource for which statistics are requested.

Possible values for Token Ring / Ethernet are:

AP_LS
Return LS statistics.

AP_PORT
Return port statistics.

For Enterprise Extender, this must be set to AP_LS.

table_type
The type of statistics information requested.

Allowed values for SDLC:

AP_STATS_TBL
Statistical information.

AP_OPER_TBL
Operational information.

For Token Ring / Ethernet , this must be set to AP_STATS_TBL.

For Enterprise Extender, this must be set to AP_STATS_TBL.

reset_stats
Specifies whether to reset the statistics when this verb completes. This parameter applies only if *table_type* is set to AP_STATS_TBL; it is ignored otherwise. Possible values are:

AP_YES
Reset the statistics; a subsequent QUERY_STATISTICS verb will contain only data gathered after this verb was issued.

AP_NO
Do not reset the statistics; the data on this verb will be included in the data returned to a subsequent QUERY_STATISTICS verb.

dlc_type
Type of the DLC. Possible values are:

AP_SDLC
Synchronous data link control

AP_TR
Token Ring

AP_ETHERNET
Ethernet

AP_X25
X.25 packet switching

AP_IP
Enterprise Extender (also known as HPR/IP)

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

dlc_type
Type of DLC for which statistics information is being returned. Possible values are:

AP_SDLC
SDLC

AP_X25
QLLC

AP_TR
Token Ring

AP_ETHERNET
Ethernet

AP_IP
Enterprise Extender (also known as HPR/IP)

statistics
Current statistics for the link station or port. This string is replaced by the appropriate structure for the DLC type. The parameters in the structure are described below.

mux_info.dlc_type*, *mux_info.card_type*, *mux_info.adapter_number
Streams configuration information for the DLC. For more information about these parameters, see “DEFINE_DLC” on page 77.

LS statistics for SDLC:

sdl_ls_stats_table.index
The index value used internally by CS Linux to identify the port that owns this LS.

sdl_ls_stats_table.address
The poll address of the secondary link station.

sdl_ls_stats_table.blus_in
The total number of basic link units (frames) received from the adjacent link station.

sdl_ls_stats_table.blus_out
The total number of basic link units (frames) transmitted to the adjacent link station.

sdl_ls_stats_table.octets_in
The total number of bytes (not including FCSs) received from the adjacent link station.

sdl_ls_stats_table.octets_out
The total number of bytes (not including FCSs) transmitted to the adjacent link station.

sdl_ls_stats_table.polls_out
Total number of polls sent to the adjacent link station.

sdl_ls_stats_table.poll_rsps_out
Total number of polls responded to by the adjacent link station.

sdl_ls_stats_table.local_busies
Total number of times the local link station has entered busy state (RNR).

sdl_ls_stats_table.remote_busies
Total number of times the remote link station has entered busy state (RNR).

sdl_ls_stats_table.iframes_in
The total number of I-frames received from the adjacent link station (including retries and out-of-order frames).

sdl_ls_stats_table.iframes_out
The total number of I-frames transmitted to the adjacent link station (including retries and out-of-order frames).

sdl_ls_stats_table.retransmits_in
The total number of retransmissions of I-frames received.

sdl_ls_stats_table.retransmits_out
The total number of retransmissions of I-frames to the adjacent link station.

sdl_ls_stats_table.ioctets_in
The total number of bytes in I-frames received from the adjacent link station.

sdl_ls_stats_table.ioctets_out
The total number of bytes in I-frames transmitted to the adjacent link station.

sdl_ls_stats_table.xids_in
The total number of XID frames received from the adjacent link station.

sdl_ls_stats_table.xids_out
The total number of XID frames transmitted to the adjacent link station.

sdl_ls_stats_table.tests_in
The total number of TEST frames, commands, or responses received from the adjacent link station.

sdl_ls_stats_table.tests_out
The total number of TEST frames, commands, or responses transmitted to the adjacent link station.

sdl_ls_stats_table.rejs_in
The total number of REJ frames received from the adjacent link station.

sdl_ls_stats_table.rejs_out
The total number of REJ frames transmitted to the adjacent link station.

sdl_ls_stats_table.frmrs_in
The total number of Frame Reject frames received from the adjacent link station.

sdl_ls_stats_table.frmrs_out
The total number of Frame Reject frames transmitted to the adjacent link station.

sdl_ls_stats_table.sims_in
The total number of Set Initialization Mode frames received from the adjacent link station.

sdl_ls_stats_table.sims_out
The total number of Set Initialization Mode frames transmitted to the adjacent link station.

sdl_ls_stats_table.rims_in
The total number of Request Initialization Mode frames received from the adjacent link station.

sdl_ls_stats_table.rims_out
The total number of Request Initialization Mode frames transmitted to the adjacent link station.

sdl_ls_stats_table.snrm_in
The total number of SNRM frames received.

sdl_ls_stats_table.snrm_out
The total number of SNRM frames sent.

LS operational information for SDLC:

sdl_ls_oper_table.index
The index value used internally by CS Linux to identify the port that owns this LS.

sdl_ls_stats_table.address
The poll address of the secondary link station.

sdl_ls_stats_table.role
The link role of the LS. Possible values are:

SDL_MIB_PRIMARY
Primary

SDL_MIB_SECONDARY
Secondary

SDL_MIB_NEGOTIABLE
Negotiable

sdl_ls_stats_table.state
An internal value indicating the processing state of the LS software (for use by support personnel).

sdl_ls_stats_table.maxdata
The current maximum PDU size allowed for the logical link (the size includes the TH and RH). For a switched line, this value may be negotiated during XID exchange.

sdl_ls_stats_table.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.

sdl_ls_stats_table.maxin
The maximum number of frames that the LS can receive before it must send an acknowledgment.

sdl_ls_stats_table.maxout
The maximum number of frames that the LS can send before it must wait for an acknowledgment.

sdl_ls_stats_table.modulo
The sequence number modulus for the LS. Possible values are:

SDL_MIB_EIGHT
8

SDL_MIB_ONETWENTYEIGHT
128

sdl_ls_stats_table.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).

sdl_ls_stats_table.retries_t
The timeout between retransmissions of a retry sequence.

sdl_ls_stats_table.retries_n
The number of times that the LS attempts to retransmit a retry sequence.

sdl_ls_stats_table.rnrlimit
The maximum length of time that the adjacent LS can remain in RNR state before the local LS considers it to be inoperative.

sdl_ls_stats_table.datmode
The communications mode with the adjacent LS. Possible values are:

SDL_MIB_HALF
Two-way alternate (half-duplex)

SDL_MIB_FULL
Two-way simultaneous (full-duplex)

sdl_ls_stats_table.last_fail_ctrl_in
The control field from the last frame received before the last failure. If the LS has not failed, this field is set to zeros.

sdl_ls_stats_table.last_fail_ctrl_out
The control field from the last frame sent before the last failure. If the LS has not failed, this field is set to zeros.

sdl_ls_stats_table.last_fail_frmr_info
If the last failure was caused by a frame that was not valid, this parameter contains the information field from the FRMR frame. If the LS has not failed, or if the failure cause was not a frame that was not valid, this field is set to zeros.

sdl_ls_stats_table.last_fail_replyto_s
The number of times that the reply timeout expired before the last failure. If the LS has not failed, this field is set to zero.

sdl_ls_stats_table.g_poll
The group poll address for the LS. If the LS is not in a group, this field is set to zero.

sdl_ls_stats_table.sim_rim
Specifies whether the LS supports transmission of SIM and RIM control frames. Possible values are:

SDL_MIB_YES
LS supports SIM and RIM.

SDL_MIB_NOLS
does not support SIM and RIM.

sdl_ls_stats_table.xmit_rcv_cap
Specifies the LS's transmit / receive capability. Possible values are:

SDL_MIB_HALF
Half-duplex

SDL_MIB_FULL
Full-duplex

Port statistics for SDLC:

sdl_port_stats_table.index
The index value used internally by CS Linux to identify the port.

sdl_port_stats_table.dwarf_frames
The number of frames received by the port that were too short to be valid.

sdl_port_stats_table.polls_out
Total number of polls sent to adjacent link stations.

sdl_port_stats_table.poll_rsps_out
Total number of polls responded to by adjacent link stations.

sdl_port_stats_table.local_busies
Total number of times the local link station has entered busy state (RNR).

sdl_port_stats_table.remote_busies
Total number of times remote link stations have entered busy state (RNR).

sdl_port_stats_table.iframes_in
The total number of I-frames received from adjacent link stations (including retries and out-of-order frames).

sdl_port_stats_table.iframes_out
The total number of I-frames transmitted to adjacent link stations (including retries and out-of-order frames).

sdl_port_stats_table.octets_in
The total number of bytes (not including FCSs) received from adjacent link stations.

sdl_port_stats_table.octets_out
The total number of bytes (not including FCSs) transmitted to adjacent link stations.

sdl_port_stats_table.protocol_errs
The number of times that CS Linux has deactivated an LS using this port because a frame received from the adjacent link station contained a protocol error.

sdl_port_stats_table.activity_to_s
The number of times that CS Linux has deactivated an LS using this port because there was no activity on the link.

sdl_port_stats_table.rnrlimit_s
The number of times that CS Linux has deactivated an LS using this port because the Remote Busy timer expired.

sdl_port_stats_table.retries_exps
The number of times that CS Linux has deactivated an LS using this port because a retry sequence has been exhausted.

sdl_port_stats_table.retransmits_in
The total number of retransmitted I-frames received from adjacent link stations.

sdl_port_stats_table.retransmits_out
The total number of retransmissions of I-frames to adjacent link stations.

Port operational information for SDLC:

sdl_port_oper_table.index
The index value used internally by CS Linux to identify the port.

sdl_port_oper_table.role
The link role of the port. Possible values are:

SDL_MIB_PRIMARY
Primary

SDL_MIB_SECONDARY
Secondary

SDL_MIB_NEGOTIABLE
Negotiable

sdl_port_oper_table.type
Specifies whether the port is operating as though connected to a leased or switched line. Possible values are:

SDL_MIB_LEASED

SDL_MIB_SWITCHED

sdl_port_oper_table.topology
Specifies whether the port can operate in a multipoint topology. Possible values are:

SDL_MIB_POINT_TO_POINT
Port can operate only as point-to-point.

SDL_MIB_MULTIPOINT
Port can operate as multipoint.

sdl_port_oper_table.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 zero indicates no timeout; the line remains connected regardless of inactivity. This parameter applies only for a switched link; its value is undefined for a leased link.

sdl_port_oper_table.pause
The length of time that the primary station waits between successive cycles of polling secondary stations. This parameter applies only if the LS role is primary; its value is undefined if the LS role is secondary.

sdl_port_oper_table.slow_poll_method
The method used for periodically polling failed secondary link stations. This is set to SDL_MIB_POLLPAUSE.

sdl_port_oper_table.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.

LS statistics for Token Ring, Ethernet :

llc2_ls_stats.local_mac
The MAC address of the local link station.

llc2_ls_stats.local_sap
The SAP address of the local link station.

llc2_ls_stats.remote_mac
The MAC address of the remote link station.

llc2_ls_stats.remote_sap
The SAP address of the remote link station.

llc2_ls_stats.rif_len
Length of the Routing Information Field data. This parameter is used only for Token Ring; it is reserved for other DLC types.

llc2_ls_stats.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 16-bit numbers in local format; the first 12 bits of each number specify the ring number, and the last 4 bits specify the bridge number.

llc2_ls_stats.ls_fsm
An internal value indicating the processing state of the LS software (for use by support personnel).

llc2_ls_stats.mac_type
The network type determined during LS activation. Possible values are:

LLC_DIX
DIX

LLC2_802_3
802.3

LLC2_802_3_DIX
Not yet determined (either 802.3 or DIX). This will change to one of the above values when the adjacent station first responds to a frame in one of these formats.

LLC2_TOKEN_RING
Token Ring

llc2_ls_stats.max_btu_size
Maximum BTU size determined during LS activation.

llc2_ls_stats.send_window
Number of I-frames the local station can send to the adjacent station before it must wait for a response.

llc2_ls_stats.receive_window
Number of I-frames the adjacent station can send to the local station before it must wait for a response.

llc2_ls_stats.t1_expiry_count
Number of times the adjacent station failed to respond within the *t1_timeout* (acknowledgment timeout) period.

llc2_ls_stats.t2_expiry_count
Number of times the *t2_timeout* period expired before a frame that could carry the required reply bits was queued.

llc2_ls_stats.remote_busy
Number of times the local station entered remote busy state because of an RNR frame from the adjacent station.

llc2_ls_stats.local_busy
Number of times the local station sent an RNR frame to the adjacent station on entering local busy state.

llc2_ls_stats.i_frames_sent
Number of I-frames sent.

llc2_ls_stats.i_bytes_sent
Number of data bytes in the I-frames sent.

llc2_ls_stats.i_frames_rcvd
Number of I-frames received.

llc2_ls_stats.i_bytes_rcvd
Number of data bytes in the I-frames received.

llc2_ls_stats.i_frames_rjctd
Number of I-frames rejected.

llc2_ls_stats.i_bytes_rjctd
Number of data bytes in the I-frames rejected.

llc2_ls_stats.i_frames_rexmit
Number of I-frames retransmitted.

llc2_ls_stats.i_bytes_rexmit
Number of data bytes in the I-frames retransmitted.

llc2_ls_stats.rej_frames_sent
Number of REJ frames sent to request retransmission of one or more I-frames.

llc2_ls_stats.rej_frames_rcvd
Number of REJ frames received requesting retransmission of one or more I-frames.

llc2_ls_stats.xid_frames_sent
Number of XID frames sent.

llc2_ls_stats.xid_frames_rcvd
Number of XID frames received.

llc2_ls_stats.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.

llc2_ls_stats.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.

llc2_ls_stats.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.

llc2_ls_stats.rej_timeout
Reject timeout: the time in seconds within which a response must be received for a REJ frame sent to the adjacent link station.

llc2_ls_stats.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.

llc2_ls_stats.idle_timeout
Idle timeout: 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.

llc2_ls_stats.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.

Port statistics for Token Ring, Ethernet:

llc2_port_stats.time_secs, time_ms
The time (in seconds and milliseconds) from when the SNA software was started to when the LLC2 component received the port activation request.

llc2_port_stats.mac_addr
MAC address of the port, determined during port activation.

llc2_port_stats.max_btu_size
Maximum BTU size, determined during port activation.

llc2_port_stats.ls_count
Number of link stations currently using the port. This includes stations for which XIDs have been sent but SABME has not yet been sent.

llc2_port_stats.ui_frames_sent
Number of Type I frames (UI, TEST, and XID) issued on this port.

llc2_port_stats.ui_frames_rcvd
Number of Type I frames (UI, TEST, and XID) received on this port.

device_stats.line_error
Total number of line errors.

device_stats.internal_error
Total number of internal errors.

device_stats.burst_error
Total number of burst errors.

device_stats.ari_fci_error
Total number of address recognized / frame copied bits errors.

device_stats.end_delim
Total number of frame delimiter errors.

device_stats.lost_frame
Total number of lost frame errors.

device_stats.rcv_cngstn
Total number of receiver congestion errors.

device_stats.frm_cpy_err
Total number of frame copied errors.

device_stats.freq_err
Total number of frequency errors.

device_stats.token_err
Total number of token errors.

device_stats.crc_err
Total number of Cyclic Redundancy Check errors.

device_stats.xmit_err
Total number of transmit errors.

device_stats.collision_err
Total number of collision errors.

LS statistics for Enterprise Extender:

udp_ls_stats_table.udp_low_out
The number of UDP datagrams sent that contained low priority APPN data.

udp_ls_stats_table.udp_med_out
The number of UDP datagrams sent that contained medium priority APPN data.

udp_ls_stats_table.udp_high_out
The number of UDP datagrams sent that contained high priority APPN data.

udp_ls_stats_table.udp_network_out
The number of UDP datagrams sent that contained network priority APPN data.

udp_ls_stats_table.udp_llc_out
The number of UDP datagrams sent that contained LLC commands.

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

リンク名の入力
指定された名前パラメーターは有効な LS 名ではありませんでした。

ポートフォリオ・ ポート名
指定された名前パラメーターは有効なポート名ではありませんでした。

アプイン・ ステータス・ タイプ
stats_type パラメーターが有効な値に設定されていませんでした。

ファイル・ タイプの追加タイプ
テーブル・タイプ パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: state check

If the verb does not execute because of a state error, CS Linux returns the following parameters.

primary_rc
AP_STATE_CHECK

secondary_rc
Possible values are:

AP_LINK_DEACTIVATED
The specified link is not currently active.

AP_PORT_DEACTIVATED
The specified port is not currently active.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_STATE_CHECK, which are common to all NOF verbs.

Returned parameters: function not supported

If the verb does not execute because the DLC type does not support returning statistics information, CS Linux returns the following parameter:

primary_rc
AP_FUNCTION_NOT_SUPPORTED

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

QUERY_TN3270_ACCESS_DEF

QUERY_TN3270_ACCESS_DEF returns information about TN3270 users on other computers that can use the TN server feature of CS 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.)

This verb can return either summary or detailed information, about a single user or multiple users, depending on the options used.

VCB structure

```
typedef struct query_tn3270_access_def
{
   AP_UINT16        opcode;                /* verb operation code          */
   unsigned char    reserv2;               /* reserved                     */
   unsigned char    format;                /* reserved                     */
   AP_UINT16        primary_rc;            /* primary return code          */
   AP_UINT32        secondary_rc;          /* secondary return code        */
   unsigned char    *buf_ptr;              /* pointer to buffer            */
   AP_UINT32        buf_size;              /* buffer size                  */
   AP_UINT32        total_buf_size;        /* total buffer size required   */
   AP_UINT16        num_entries;           /* number of entries            */
   AP_UINT16        total_num_entries;     /* total number of entries      */
   unsigned char    list_options;          /* listing options              */
   unsigned char    reserv3;               /* reserved                     */
   AP_UINT16        default_record;        /* start with DEFAULT record?   */
   unsigned char    client_address[256];   /* address of TN3270 user       */
   AP_UINT16        port_number;           /* TCP/IP port to access server */
   AP_UINT32        num_init_sessions;     /* number of sessions for first */
                                           /* user when starting in middle */
   AP_UINT32        num_last_sessions;     /* number of sessions on last   */
                                           /*  detail overlay if last user */
                                           /*  is incomplete               */
   unsigned char    last_user_incomplete;  /* set to AP_YES if session     */
                                           /*  data for last user incomplete*/
   unsigned char    reserv4[11];           /* Reserved                     */
} QUERY_TN3270_ACCESS_DEF;
```

```
typedef struct tn3270_access_summary
{
   AP_UINT16        overlay_size;          /* overlay size                 */
   AP_UINT16        default_record;        /* is this the DEFAULT record?  */
   unsigned char    client_address[256];   /* address of TN3270 user       */
   AP_UINT16        address_format;        /* Format of client address     */
   unsigned char    reserv3[6];            /* Reserved                     */
} TN3270_ACCESS_SUMMARY;
```

```
typedef struct tn3270_access_detail
{
  AP_UINT16              overlay_size;         /* overlay size              */
  AP_UINT16              sub_overlay_offset;   /* offset to first sess struct*/
  AP_UINT16              default_record;       /* is this the DEFAULT record?*/
  unsigned char          client_address[256];  /* address of TN3270 user    */
  AP_UINT32              num_filtered_sessions;/* num sess returned for user */
  unsigned char          reserv3[4];           /* Reserved                  */
  TN3270_ACCESS_DEF_DATA def_data;             /* user definition           */
} TN3270_ACCESS_DETAIL;
```

```
typedef struct tn3270_access_def_data
{
   unsigned char   description[32];     /* Description - null terminated    */
   unsigned char   reserv1[16];         /* reserved                         */
   AP_UINT16       address_format;      /* Format of client address         */
   AP_UINT32       num_sessions;        /* Number of sessions being added   */
   unsigned char   reserv3[64];         /* reserved                         */
} TN3270_ACCESS_DEF_DATA;
```

For each session, up to the number specified by the *num_sessions* parameter, the following structure is included at the end of the `def_data` structure:

```
typedef struct tn3270_session_def_data
{
   AP_UINT16           sub_overlay_size;   /* reserved                     */
   unsigned char       description[32];    /* Session description          */
   unsigned char       tn3270_support;     /* Level of TN3270 support      */
   unsigned char       allow_specific_lu;  /* Allow access to specific LUs */
   unsigned char       printer_lu_name[8]; /* Generic printer LU/pool      */
                                           /*   accessed                   */
   unsigned char       reserv1[6];         /* reserved                     */
   AP_UINT16           port_number;        /* TCP/IP port used to access   */
                                           /*  server                      */
   unsigned char       lu_name[8];         /* Generic display LU/pool      */
                                           /*  accessed                    */
   unsigned char       session_type;       /* Unused in current version    */
   unsigned char       model_override;     /* Unused in current version    */
```

```
   unsigned char       ssl_enabled;        /* Is this an SSL session?        */
   unsigned char       security_level;     /* SSL encryption strength        */
   unsigned char       cert_key_label[80]; /* Certificate key label          */
   unsigned char       listen_local_address[46];
                                           /* Local addr client connects to  */
  unsigned char        allow_ssl_timeout_to_nonssl;
                                           /* Allow non-SSL clients on SSL?  */
  unsigned char     reserv3[17];
   AP_UINT32           reserv4;            /* reserved                       */
} TN3270_SESSION_DEF_DATA;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_TN3270_ACCESS_DEF

buf_ptr
A pointer to a data buffer that CS Linux will use to return the requested information.

buf_size
Size of the supplied data buffer.

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

To request data for a specific user rather than a range, specify the value 1. To return as many entries as possible, specify zero; in this case, CS Linux will return the maximum number of entries that can be accommodated in the supplied data buffer.

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

AP_SUMMARY
Summary information only.

AP_DETAIL
Detailed information.

Combine this value using a logical OR operation with one of the following values:

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

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

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

The list is ordered by client address and then by port number for each user. For more information about how the list is ordered and how the application can obtain specific entries from it, see “List options for QUERY_* Verbs” on page 34.

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

To query the default record, which is used by any TN3270 user not explicitly identified by a TCP/IP address, specify AP_YES. In this case, the *client_address* parameter is reserved.

To query a normal TN3270 user record, specify AP_NO.

client_address
The TCP/IP address of the TN3270 user for whom information is required, or the name to be used as an index into the list of users. This parameter is ignored if *list_options* is set to AP_FIRST_IN_LIST. The address is a null-terminated ASCII string, which 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).

port_number
To return information starting with a specific session for the specified user, set this parameter to the TCP/IP port number defined for that session. To return information starting at the first session for the specified user, set this parameter to zero.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than *buf_size*.

total_num_entries
Total number of entries that could have been returned. This may be higher than *num_entries*.

num_entries
The number of entries actually returned. The last entry may be incomplete; this is indicated by the *last_user_incomplete* parameter.

num_init_sessions
If the *port_number* parameter was set to a nonzero value, so that the information for the first user in the list does not start with the user's first session, this parameter indicates the number of session structures for this user that are included in the returned data. Otherwise, this parameter is not used.

num_last_sessions
If the *last_user_incomplete* parameter indicates that the data for the last user is incomplete, this parameter indicates the number of session structures for this user that are included in the returned data. Otherwise, this parameter is not used.

last_user_incomplete
Specifies whether the information for the last user is incomplete. Possible values are:

AP_YES
The complete data for the last user was too large to fit in the data buffer. At least one session structure is included, but there are further session structures that are not included in the data buffer. The *num_last_sessions* parameter indicates how many session structures have been returned; the application can issue further verbs to obtain the remaining data.

AP_NO
The data for the last user is complete.

Each entry in the data buffer consists of the following:

tn3270_access_summary.overlay_size
The size of the returned tn3270_access_summary structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `tn3270_access_summary` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

tn3270_access_summary.default_record
Specifies whether this entry is the default record. Possible values are:

AP_YES
This is the default record. The *client_address* parameter is reserved.

AP_NO
This is a normal TN3270 user record.

tn3270_access_summary.client_address
The TCP/IP address of the TN3270 user. 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`).

tn3270_access_summary.address_format
Specifies the format of the *client_address* parameter. Possible values are:

AP_ADDRESS_IP
IP address (either IPv4 or IPv6)

AP_ADDRESS_FQN
Alias or fully qualified name

tn3270_access_detail.overlay_size
The size of the returned `tn3270_access_detail` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `tn3270_access_detail` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

tn3270_access_detail.sub_overlay_offset
The offset to the start of the first session data structure for this TN3270 access record in the data buffer.

tn3270_access_detail.default_record
Specifies whether this entry is the default record. Possible values are:

AP_YES
This is the default record. The *client_address* parameter is reserved.

AP_NO
This is a normal TN3270 user record.

tn3270_access_detail.client_address
The TCP/IP address of the TN3270 user. This is a null-terminated ASCII string, which 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).

tn3270_access_detail.num_filtered_sessions
The number of sessions returned for this user.

tn3270_access_detail.def_data
The details of the user, as defined in the configuration. This is followed by a number of session structures defining the user's sessions. The format of this information is the same as for the DEFINE_TN3270_ACCESS verb, except for the following:

- The *num_sessions* parameter in the `def_data` structure defines the total number of sessions defined for the user.
- If the *port_number* parameter was set to a nonzero value, the data for the first user will contain only the remaining session structures (starting from the requested entry), without the `def_data` structure.
- If the *last_user_incomplete* parameter is set to AP_YES, the total number of session structures returned for the last user will be as specified by the *num_last_sessions* parameter; this will be less than *num_sessions*.

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

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

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

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

QUERY_TN3270_ASSOCIATION

QUERY_TN3270_ASSOCIATION returns information about associations between display LUs and printer LUs. Associations are queried by display LU name and are returned in order of display LU name.

This verb can be used to obtain information about a specific association or about multiple associations, depending on the options used.

VCB structure

```
typedef struct query_tn3270_association
{
   AP_UINT16        opcode;               /* verb operation code           */
   unsigned char    reserv2;              /* reserved                      */
   unsigned char    format;               /* reserved                      */
   AP_UINT16        primary_rc;           /* primary return code           */
```

```
   AP_UINT32        secondary_rc;          /* secondary return code          */
   unsigned char   *buf_ptr;               /* pointer to buffer              */
   AP_UINT32        buf_size;              /* buffer size                    */
   AP_UINT32        total_buf_size;        /* total buffer size required     */
   AP_UINT16        num_entries;           /* number of entries              */
   AP_UINT16        total_num_entries;     /* total number of entries        */
   unsigned char    list_options;          /* listing options                */
   unsigned char    reserv3;               /* reserved                       */
   unsigned char    display_lu_name[8];    /* Display LU name                */
} QUERY_TN3270_ASSOCIATION;
```

```
typedef struct tn3270_association
{
   AP_UINT16      overlay_size;            /* Overlay size                   */
   unsigned char  reserv2[2];              /* reserved                       */
   unsigned char  display_lu_name[8];      /* Display LU name                */
   TN3270_ASSOCIATION_DEF_DATA def_data;   /* association definition         */
} TN3270_ASSOCIATION;
```

```
typedef struct tn3270_association_def_data
{
   unsigned char  description[32];         /* resource description           */
   unsigned char  reserve0[16];            /* reserved                       */
   unsigned char  printer_lu_name[8];      /* name of printer LU/pool        */
   unsigned char  reserv2[8];              /* reserved                       */
} TN3270_ASSOCIATION_DEF_DATA;
```

Data is returned in the form of `tn3270_association` structures.

Supplied parameters

The application supplies the following parameters:

opcode
`AP_QUERY_TN3270_ASSOCIATION`

buf_ptr
A pointer to a data buffer that CS Linux will use to return the requested information.

buf_size
Size of the supplied data buffer.

num_entries
Maximum number of associations for which data should be returned. To request data for a specific association rather than a range, specify the value 1. To return as many entries as possible, specify 0; in this case, CS Linux will return the maximum number of entries that can be accommodated in the supplied data buffer.

list_options
The position in the list of associations from which CS Linux begins to return data. Specify one of the following values:

AP_FIRST_IN_LIST
Start at the first entry in the list.

AP_LIST_INCLUSIVE
Start at the entry specified by the *display_lu_name* parameter.

AP_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 EBCDIC string padded on the right with EBCDIC spaces. This parameter is ignored if *list_options* is set to `AP_FIRST_IN_LIST`.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than the value supplied for the *buf_size* parameter.

num_entries
The number of entries actually returned.

total_num_entries
Total number of entries that could have been returned. This may be higher than the value supplied for the *num_entries* parameter.

Each entry in the data buffer consists of the following:

tn3270_association.overlay_size
The size of the returned `tn3270_association` structure (and therefore the offset to the start of the next entry in the data buffer).

When your application needs to go through the returned buffer to find each `tn3270_association` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

tn3270_association.display_lu_name
Name of the display LU associated with the printer LU specified by the *association.printer_lu_name* parameter. This is an EBCDIC string padded on the right with EBCDIC spaces.

tn3270_association_def_data.description
A null-terminated text string that describe the association, as specified in the definition of the association.

tn3270_association_def_data.printer_lu_name
Name of the printer LU associated with the display LU specified by the *association.display_lu_name* parameter. This is an EBCDIC string padded on the right with EBCDIC spaces.

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

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

AP_INVALID_LU_NAME
Indicates one of the following:

- The *list_options* parameter was set to AP_LIST_FROM_NEXT, but the display LU name was not a valid EBCDIC string.
- The *list_options* parameter was set to AP_LIST_INCLUSIVE, but the display LU name either was not a valid EBCDIC string or did not correspond to an existing association record.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

照会に使用されます。デフォルト

QUERY_TN3270_DEFAULTS は、すべてのクライアント・セッションで使用される TN3270 パラメーターに関する情報を戻します。

Secure Sockets Layer (SSL) クライアント認証を使用しており、外部 LDAP サーバー上の証明書失効リストに対してクライアントを検査する場合は、 QUERY_TN3270_SSL_LDAP verb を使用して、このサーバーにアクセスする方法の詳細を戻します。

VCB structure

```
typedef struct query_tn3270_defaults
{
   AP_UINT16        opcode;               /* verb operation code            */
   unsigned char    reserv2;              /* reserved                       */
   unsigned char    format;               /* reserved                       */
   AP_UINT16        primary_rc;           /* primary return code            */
   AP_UINT32        secondary_rc;         /* secondary return code          */
   TN3270_DEFAULTS_DEF_DATA def_data;     /* TN3270 defaults                */
} QUERY_TN3270_DEFAULTS;
```

```
typedef struct tn3270_defaults_def_data
{
   AP_UINT16        force_responses;      /* force printer responses?       */
   AP_UINT16        keepalive_method;     /* method for sending keep-alives */
   AP_UINT32        keepalive_interval;   /* interval between keep-alives   */
   unsigned char    reserv2[32];          /* reserved                       */
} TN3270_DEFAULTS_DEF_DATA;
```

Supplied parameters

The application supplies the following parameter:

opcode
AP_QUERY_TN3270_DEFAULTS

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

def_data.force_responses
Controls client responses on printer sessions. Possible values are:

AP_YES
Requests definite responses.

AP_NO
Request responses matching SNA traffic.

def_data.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:

AP_NONE
Do not send keep-alive messages.

AP_TN3270_NOP
Send Telnet NOP messages.

AP_TN3270_TM
Send Telnet DO TIMING-MARK messages.

def_data.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 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).

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

QUERY_TN3270_EXPRESS_LOGON

QUERY_TN3270_EXPRESS_LOGON returns information about the TN3270 Express Logon feature. This feature means that TN3270 client users who connect to CS 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.

VCB 構造体

```
typedef struct query_tn3270_express_logon
{
   AP_UINT16        opcode;               /* verb operation code           */
   unsigned char    reserv2;              /* reserved                      */
   unsigned char    format;               /* reserved                      */
   AP_UINT16        primary_rc;           /* primary return code           */
   AP_UINT32        secondary_rc;         /* secondary return code         */
   unsigned char    dcas_server[256];     /* IP hostname of DCAS server    */
   AP_UINT16        dcas_port;            /* port number to access server  */
   unsigned char    enabled;              /* is Express Logon enabled?     */
   unsigned char    reserv3[33];          /* reserved                      */
} QUERY_TN3270_EXPRESS_LOGON;
```

Supplied parameters

The application supplies the following parameter:

opcode
AP_QUERY_TN3270_EXPRESS_LOGON

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

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:

AP_YES
The function is enabled, so TN3270 clients can access the host without needing to specify a user ID and password.

AP_NO
The function is not enabled, so TN3270 clients must specify a user ID and password.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

照会に対する照会の LDAP マップ・マップ

QUERY_TN3270_SSL_LDAP は、 Secure Sockets Layer (SSL) クライアント認証機能で使用するための証明書取り消しリストへのアクセス方法に関する情報を戻します。 この情報は、 DEFINE_TN3270_SSL_LDAP verb を使用して指定されました。

VCB 構造体

```
typedef struct query_tn3270_ssl_ldap
{
   AP_UINT16        opcode;               /* verb operation code            */
   unsigned char    reserv2;              /* reserved                       */
   unsigned char    format;               /* reserved                       */
   AP_UINT16        primary_rc;           /* primary return code            */
   AP_UINT32        secondary_rc;         /* secondary return code          */
   unsigned char   *buf_ptr;              /* pointer to buffer              */
   AP_UINT32        buf_size;             /* buffer size                    */
   AP_UINT32        total_buf_size;       /* total buffer size required     */
   AP_UINT16        num_entries;          /* reserved                       */
   AP_UINT16        total_num_entries;    /* reserved                       */
   unsigned char    list_options;         /* reserved                       */
   unsigned char    reserv3;              /* reserved                       */
} QUERY_TN3270_SSL_LDAP;
```

```
typedef struct tn3270_ssl_ldap_def_data
{
   AP_UINT16           overlay_size;       /* reserved                       */
   unsigned char       auth_type;          /* type of authorization checking */
   unsigned char       reserv1;            /* reserved                       */
   unsigned char       ldap_addr[256];     /* address of LDAP server         */
   AP_UINT16           ldap_port;          /* port number to access server   */
   unsigned char       ldap_user[1024];    /* user ID on LDAP server         */
   unsigned char       ldap_password[128]; /* password on LDAP server        */
   unsigned char       reserv2[256];       /* reserved                       */
} TN3270_SSL_LDAP_DEF_DATA;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
照会の数の最大値の 1 つのマップ（ _T ）

buf_ptr
CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return the complete information. This may be higher than the value supplied for the *buf_size* parameter.

The following information is returned in the data buffer:

def_data.auth_type
Specifies the type of authorization checking performed by the TN Server or TN Redirector. Possible values are:

AP_LOCAL_ONLY
The server checks client certificates locally, but does not use an external certificate revocation list. The parameters *ldap_addr* - *ldap_password* are reserved.

AP_LOCAL_X500
The server checks certificates locally, and also checks against an external certificate revocation list. The remaining parameters in this data structure specify the location of this list.

def_data.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`).

def_data.ldap_port
The TCP/IP port number used to access the LDAP server. The range is 0-65535.

def_data.ldap_user
The user name used to access the certificate revocation list on the LDAP server.

def_data.ldap_password
The password used to access the certificate revocation list on the LDAP server.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

QUERY_TN_REDIRECT_DEF

QUERY_TN_REDIRECT_DEF returns information about Telnet clients on other computers that can use the TN Redirector feature of CS Linux to access a host. It can return either summary or detailed information, about a single user or multiple users, depending on the options used.

VCB structure

```
typedef struct query_tn_redirect_def
{
   AP_UINT16        opcode;                /* verb operation code          */
   unsigned char    reserv2;               /* reserved                     */
   unsigned char    format;                /* reserved                     */
   AP_UINT16        primary_rc;            /* primary return code          */
   AP_UINT32        secondary_rc;          /* secondary return code        */
   unsigned char    *buf_ptr;              /* pointer to buffer            */
   AP_UINT32        buf_size;              /* buffer size                  */
   AP_UINT32        total_buf_size;        /* total buffer size required   */
   AP_UINT16        num_entries;           /* number of entries            */
   AP_UINT16        total_num_entries;     /* total number of entries      */
   unsigned char    list_options;          /* listing options              */
   unsigned char    reserv3[3];            /* reserved                     */
  TN_REDIRECT_ADDRESS addr;               /* Uniquely defines record      */
} QUERY_TN_REDIRECT_DEF;
```

```
typedef struct tn_redirect_data
{
   AP_UINT16             overlay_size;     /* overlay size                 */
   unsigned char         reserv1[2];       /* Reserved                     */
   TN_REDIRECT_ADDRESS   addr;             /* addressing information       */
   TN_REDIRECT_DEF_DATA  def_data;         /* definitions for the client   */
} TN_REDIRECT_DATA;
```

```
typedef struct tn_redirect_address
{
   AP_UINT16      default_record;    /* Is this the default record ?      */
   unsigned char  address_format;    /* IP address or fully-qualified name*/
   unsigned char  client_address[256]; /* Client address                  */
   AP_UINT16      port_number;       /* Port number that client connects on */
   unsigned char  listen_local_address[46];
                                     /* Local addr client connects to     */
   unsigned char  reserved[34];      /* reserved                          */
} TN_REDIRECT_ADDRESS;
```

```
typedef struct tn_redirect_def_data
{
   unsigned char  description[32];       /* Description - null terminated */
   unsigned char  reserve0[16];          /* Reserved                      */
   unsigned char  cli_ssl_enabled;       /* Is the client session SSL?    */
   unsigned char  host_ssl_enabled;      /* Is the host session SSL?      */
   unsigned char  host_address_format;   /* Type of IP address for the host */
   unsigned char  reserv1;               /* Reserved                      */
   unsigned char  host_address[256];     /* Host address                  */
   AP_UINT16      host_port_number;      /* Port number to connect to host */
   unsigned char  cli_conn_security_level;      /* SSL encryption strength */
   unsigned char  serv_conn_security_level;     /* SSL encryption strength */
   unsigned char  cli_conn_cert_key_label[80];  /* Key label for certificate */
   unsigned char  serv_conn_cert_key_label[80]; /* Key label for certificate */
   unsigned char  reserved[46];                 /* Reserved                 */
} TN_REDIRECT_DEF_DATA;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
照会のリダイレクト （ 定義 ）

buf_ptr
CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

num_entries
データが戻されるユーザーの最大数。 特定の範囲ではなく、特定のユーザーのデータを要求するには、 1 という値を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、 CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータの戻りを開始する必要のあるリスト内の位置、および各項目に必要な情報のレベル。 次の値のいずれかを指定してください。

リストの最初のリスト （ _R ）
リスト内の最初のユーザーから開始します。

リストを含む （ 包括的 ）
提供されたクライアント・アドレス指定情報で指定されたエントリーから開始します。

次への AP_LIST_FROM_NEXT
提供されたクライアント・アドレス指定情報によって指定されたエントリーの直後のエントリーから開始します。

リストはクライアント・アドレス順に並べられます。 リストの順序とアプリケーションでの特定のエントリーの取得方法について詳しくは、 34 ページの『List options for QUERY_* Verbs』を参照してください。

アドレス
情報が必要な Telnet クライアント、またはユーザーのリストへの索引として使用されるユーザーのアドレッシング情報を指定します。 このデータ構造の内容について詳しくは、 188 ページの『定義済みのリダイレクト』を参照してください。

list_options が **リストの最初のリスト （ _R ）**に設定されている場合、この構造体の情報は無視されます。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. This may be higher than *buf_size*.

total_num_entries
Total number of entries that could have been returned. This may be higher than *num_entries*.

num_entries
The number of entries actually returned.

tn_redirect_data.overlay_size
The size of the returned `tn_redirect_data` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `tn_redirect_data` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

tn_redirect_data.addr
Specifies addressing information for the Telnet client. For more information about the contents of this data structure, see “定義済みのリダイレクト” on page 188.

tn_redirect_data.def_data
Specifies definitions for the Telnet client. For more information about the contents of this data structure, see “定義済みのリダイレクト” on page 188.

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

ユーザー・ アドレスの追加 （ _E ）
list_options パラメーターが **リストを含む** （ **包括的**)に設定されましたが、指定されたアドレッシング情報が、定義済みの TN リダイレクター・ユーザーと一致しませんでした。

ファイルの追加リスト・ オプション
list_options パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

QUERY_TN_SERVER_TRACE

This verb returns information about the current tracing options for the CS Linux TN server feature.

This verb must be issued to a running node.

VCB structure

```
typedef struct query_tn_server_trace
{
   AP_UINT16        opcode;                    /* verb operation code     */
   unsigned char    reserv2;                   /* reserved                */
   unsigned char    format;                    /* reserved                */
   AP_UINT16        primary_rc;                /* primary return code     */
   AP_UINT32        secondary_rc;              /* secondary return code   */
   AP_UINT16        trace_flags;               /* trace flags             */
   unsigned char    reserv3[6];                /* Reserved                */
} QUERY_TN_SERVER_TRACE;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
照会サーバー・ サーバー・ トレースの追加

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

secondary_rc
Not used.

trace_flags
The types of tracing currently active.

If no tracing is active, or if tracing of all types is active, this is one of the following values:

AP_TN_SERVER_NO_TRACE
No tracing.

AP_TN_SERVER_ALL_TRACE
Tracing of all types.

If tracing is being used on specific interfaces, this parameter is set to one or more values from the list below, combined using a logical OR operation.

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

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

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

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

照会 _TP

QUERY_TP は、ローカル LU を現在使用している TP に関する情報を戻します。 この verb は、使用されるオプションに応じて、特定の TP に関する情報、または複数の TP に関する情報を取得するために使用できます。 この verb は、 TP の定義についてではなく、 TP の現在の使用状況に関する情報を戻します。 QUERY_TP_DEFINITION を使用して、 TP の定義を取得します。

この verb は実行中のノードに対して発行する必要があります

VCB structure

```
typedef struct query_tp
{
  AP_UINT16        opcode;                 /* Verb operation code        */
  unsigned char    reserv2;                /* reserved                   */
  unsigned char    format;                 /* reserved                   */
  AP_UINT16        primary_rc;             /* Primary return code        */
  AP_UINT32        secondary_rc;           /* Secondary return code      */
  unsigned char    *buf_ptr;               /* pointer to buffer          */
  AP_UINT32        buf_size;               /* buffer size                */
  AP_UINT32        total_buf_size;         /* total buffer size required */
  AP_UINT16        num_entries;            /* number of entries          */
  AP_UINT16        total_num_entries;      /* total number of entries    */
  unsigned char    list_options;           /* listing options            */
  unsigned char    reserv3;                /* reserved                   */
  unsigned char    lu_name[8];             /* LU name                    */
  unsigned char    lu_alias[8];            /* LU alias                   */
  unsigned char    tp_name[64];            /* TP name                    */
} QUERY_TP;
```

```
typedef struct tp_data
{
  AP_UINT16        overlay_size;           /* size of returned entry     */
  unsigned char    tp_name[64];            /* TP name                    */
  unsigned char    description[32];         /* resource description       */
  unsigned char    reserv1[16];            /* reserved                   */
  AP_UINT16        instance_limit;         /* maximum instance count     */
  AP_UINT16        instance_count;         /* current instance count     */
  AP_UINT16        locally_started_count;  /* locally started instance   */
                                           /*  count                     */
  AP_UINT16        remotely_started_count; /* remotely started instance  */
```

```
                                                /*  count                      */
  unsigned char    reserva[20];                 /* reserved                    */
} TP_DATA;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_TP

buf_ptr
A pointer to a data buffer that CS Linux will use to return the requested information.

buf_size
Size of the supplied data buffer.

num_entries
Maximum number of TPs for which data should be returned. To request data for a specific TP rather than a range, specify the value 1. To return as many entries as possible, specify zero; in this case, CS Linux will return the maximum number of entries that can be accommodated in the supplied data buffer.

list_options
The position in the list of TPs from which CS Linux should begin to return data. Possible values are:

AP_FIRST_IN_LIST
Start at the first entry in the list.

AP_LIST_INCLUSIVE
Start at the entry specified by the combination of LU name and TP name.

AP_LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the combination of LU name and TP name.

For more information about how the list is ordered and how the application can obtain specific entries from it, see “List options for QUERY_* Verbs” on page 34.

lu_name
LU name. This is an 8-byte type-A EBCDIC string, padded on the right with spaces if the name is shorter than 8 characters. To specify that the LU is identified by its alias rather than its LU name, set this parameter to 8 binary zeros and specify the LU alias in the following parameter. To specify the LU associated with the local CP (the default LU), set both *lu_name* and *lu_alias* to binary zeros.

lu_alias
Locally defined LU alias. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes. This parameter is used only if *lu_name* is set to 8 binary zeros; it is ignored otherwise. To specify the LU associated with the local CP (the default LU), set both *lu_name* and *lu_alias* to binary zeros.

tp_name
TP name. This is a 64-byte string, padded on the right with spaces if the name is shorter than 64 characters. This value is ignored if *list_options* is set to AP_FIRST_IN_LIST.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

tp_data.overlay_size
The size of the returned `tp_data` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `tp_data` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

tp_data.tp_name
TP name. This is a 64-byte string, padded on the right with spaces if the name is shorter than 64 characters.

tp_data.description
A null-terminated text string describing the TP, as specified in the definition of the TP.

tp_data.instance_limit
Maximum number of concurrently active instances of the specified TP.

tp_data.instance_count
Number of instances of the specified TP that are currently active.

tp_data.locally_started_count
Number of instances of the TP that have been started locally (by the TP issuing a TP_STARTED verb).

tp_data.remotely_started_count
Number of instances of the TP that have been started remotely (by a received Attach request).

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
`AP_PARAMETER_CHECK`

secondary_rc
Possible values are:

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

AP_INVALID_LU_ALIAS
The supplied *lu_alias* parameter was not valid.

AP_INVALID_LU_NAME
The supplied *lu_name* parameter was not valid.

AP_INVALID_TP_NAME
The *list_options* parameter was set to `AP_LIST_INCLUSIVE` to list all entries starting from the supplied name, but the *tp_name* parameter was not valid.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with `AP_PARAMETER_CHECK`, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

QUERY_TP_DEFINITION

QUERY_TP_DEFINITION returns information about TPs defined on the CS Linux system. This verb can be used to obtain information about a specific TP or about multiple TPs, depending on the options used. It returns information about the definition of the TPs, not about their current usage; use QUERY_TP to obtain the usage information.

VCB structure

```
typedef struct query_tp_definition
{
   AP_UINT16        opcode;                /* Verb operation code           */
   unsigned char    reserv2;               /* reserved                      */
   unsigned char    format;                /* reserved                      */
   AP_UINT16        primary_rc;            /* Primary return code           */
   AP_UINT32        secondary_rc;          /* Secondary return code         */
   unsigned char    *buf_ptr;              /* pointer to buffer             */
   AP_UINT32        buf_size;              /* buffer size                   */
   AP_UINT32        total_buf_size;        /* total buffer size required    */
   AP_UINT16        num_entries;           /* number of entries             */
   AP_UINT16        total_num_entries;     /* total number of entries       */
   unsigned char    list_options;          /* listing options               */
   unsigned char    reserv3;               /* reserved                      */
   unsigned char    tp_name[64];           /* TP name                       */
} QUERY_TP_DEFINITION;
```

```
typedef struct tp_def_summary
{
   AP_UINT16        overlay_size;          /* size of returned entry        */
   unsigned char    tp_name[64];           /* TP name                       */
   unsigned char    description[32];       /* resource description          */
   unsigned char    reserv1[16];           /* reserved                      */
} TP_DEF_SUMMARY;
```

```
typedef struct tp_def_detail
{
   AP_UINT16        overlay_size;          /* size of returned entry        */
   unsigned char    tp_name[64];           /* TP name                       */
   TP_CHARS         tp_chars;              /* TP characteristics            */
} TP_DEF_DETAIL;
```

```
typedef struct tp_chars
{
   unsigned char    description[32];        /* resource description          */
   unsigned char    security_list_name[14]; /* security access list name     */
   unsigned char    reserv1[2];             /* reserved                      */
   unsigned char    conv_type;              /* conversation type             */
   unsigned char    security_rqd;           /* security support              */
   unsigned char    sync_level;             /* synchronization level support */
   unsigned char    dynamic_load;           /* dynamic load                  */
   unsigned char    enabled;                /* is the TP enabled?            */
   unsigned char    pip_allowed;            /* program initialization        */
                                            /*  parameters supported         */
   unsigned char    reserv3[10];            /* reserved                      */
   AP_UINT16        tp_instance_limit;      /* limit on currently active TP  */
                                            /*  instances                    */
   AP_UINT16        incoming_alloc_timeout; /* incoming allocation timeout   */
   AP_UINT16        rcv_alloc_timeout;      /* receive allocation timeout    */
   AP_UINT16        tp_data_len;            /* reserved                      */
   unsigned char    tp_data[120];           /* reserved                      */
} TP_CHARS;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_TP_DEFINITION

buf_ptr
A pointer to a data buffer that CS Linux will use to return the requested information.

buf_size
Size of the supplied data buffer.

num_entries
Maximum number of TPs for which data should be returned. To request data for a specific TP rather than a range, specify the value 1. To return as many entries as possible, specify zero; in this case, CS Linux will return the maximum number of entries that can be accommodated in the supplied data buffer.

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

AP_SUMMARY
Summary information only.

AP_DETAIL
Detailed information.

Combine this value using a logical OR operation with one of the following values:

AP_FIRST_IN_LIST
Start at the first entry in the list.

AP_LIST_INCLUSIVE
Start at the entry specified by the *tp_name* parameter.

AP_LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *tp_name* parameter.

For more information about how the list is ordered and how the application can obtain specific entries from it, see "List options for QUERY_* Verbs" on page 34.

tp_name
TP name. This is a 64-byte string, padded on the right with spaces if the name is shorter than 64 characters. This parameter is ignored if *list_options* is set to AP_FIRST_IN_LIST.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

tp_def_summary.overlay_size
The size of the returned `tp_def_summary` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `tp_def_summary` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

tp_def_summary.tp_name
TP name. This is a 64-byte string, padded on the right with spaces if the name is shorter than 64 characters.

tp_def_summary.description
A null-terminated text string describing the TP, as specified in the definition of the TP.

tp_def_detail.overlay_size
The size of the returned `tp_def_detail` structure, and therefore the offset to the start of the next entry in the data buffer.

When your application needs to go through the returned buffer to find each `tp_def_detail` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

tp_def_detail.tp_name
TP name. This is a 64-byte string, padded on the right with spaces if the name is shorter than 64 characters.

tp_def_detail.tp_chars.description
A null-terminated text string describing the TP, as specified in the definition of the TP.

tp_def_detail.tp_chars.security_list_name
Name of the security access list used by this TP (defined using the DEFINE_SECURITY_ACCESS_LIST verb). This parameter restricts the TP so that only the users named in the specified list can allocate conversations with it.

If this parameter is set to 14 binary zeros, the TP is available for use by any user.

tp_def_detail.tp_chars.conv_type
Specifies the type or types of conversation supported by the TP. Possible values are:

AP_BASIC
The TP supports only basic conversations.

AP_MAPPED
The TP supports only mapped conversations.

AP_EITHER
The TP supports either basic or mapped conversations.

tp_def_detail.tp_chars.security_rqd
Specifies the level of conversation security information required to start the TP. Possible values are:

AP_YES
A user ID and password are required to start the TP.

AP_NO
No security information is required.

tp_def_detail.tp_chars.sync_level
Specifies the values of synchronization level supported by the TP. Possible values are:

AP_NONE
The TP supports only *sync_level* NONE.

AP_CONFIRM_SYNC_LEVEL
The TP supports only *sync_level* CONFIRM.

AP_EITHER
The TP supports either *sync_level* NONE or CONFIRM.

AP_SYNCPT_REQUIRED
The TP supports only *sync_level* SYNCPT (syncpoint is required).

AP_SYNCPT_NEGOTIABLE
The TP supports any of the three *sync_level* values NONE, CONFIRM, and SYNCPT.

tp_def_detail.tp_chars.dynamic_load
Specifies whether the TP can be dynamically loaded. This is set to AP_YES.

tp_def_detail.tp_chars.enabled
Specifies whether the TP can be attached successfully. Possible values are:

AP_YES
TP can be attached.

AP_NO
TP cannot be attached.

tp_def_detail.tp_chars.pip_allowed
Specifies whether the TP can receive Program Initialization Parameters (PIP). Possible values are:

AP_YES
TP can receive PIP.

AP_NO
TP cannot receive PIP.

tp_def_detail.tp_chars.duplex_support
Specifies which conversation duplex types are supported by the TP. Possible values are:

AP_HALF_DUPLEX
The TP supports half-duplex conversations only.

AP_FULL_DUPLEX
The TP supports full-duplex conversations.

AP_EITHER_DUPLEX
The TP supports both half-duplex and full-duplex conversations.

tp_def_detail.tp_chars.tp_instance_limit
Limit on the number of concurrently active TP instances.

tp_def_detail.tp_chars.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.

tp_def_detail.tp_chars.rcv_alloc_timeout
Number of seconds that a RECEIVE_ALLOCATE verb is queued waiting for an incoming Attach. The value 0 (zero) implies that there is no timeout; the RECEIVE_ALLOCATE verb will be queued indefinitely.

tp_def_detail.tp_chars.tp_data_len
Length of the implementation dependent TP data.

tp_def_detail.tp_chars.tp_data
CS Linux does not use this parameter (it is set to all zeros).

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_TP_NAME
The *list_options* parameter was set to AP_LIST_INCLUSIVE to list all entries starting from the supplied name, but the *tp_name* parameter was not valid.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

照会 TP_LOAD_INFO

QUERY_TP_LOAD_INFO は、 TP ロード情報エントリーに関する情報を戻します。 バッファーには、可変サイズの tp_load_info 構造の数が含まれています。

この verb は実行中のノードに対して発行する必要があります

VCB structure

```
typedef struct query_tp_load_info
{
  AP_UINT16               opcode;               /* Verb operation code        */
  unsigned char           reserv2;              /* reserved                   */
  unsigned char           format;               /* reserved                   */
  AP_UINT16               primary_rc;           /* Primary return code        */
  AP_UINT32               secondary_rc;         /* Secondary return code      */
  unsigned char           *buf_ptr;             /* pointer to buffer          */
  AP_UINT32               buf_size;             /* buffer size                */
  AP_UINT32               total_buf_size;       /* total buffer size required */
  AP_UINT16               num_entries;          /* number of entries          */
  AP_UINT16               total_num_entries;    /* total number of entries    */
  unsigned char           list_options;         /* listing options            */
  unsigned char           reserv3[3];           /* reserved                   */
  unsigned char           tp_name[64];          /* TP name                    */
  unsigned char           lu_alias[8];          /* LU alias                   */
} QUERY_TP_LOAD_INFO;
```

```
typedef struct tp_load_info
{
  AP_UINT16               overlay_size;         /* size of returned entry     */
  unsigned char           tp_name[64];          /* TP name                    */
  unsigned char           lu_alias[8];          /* LU alias                   */
  TP_LOAD_INFO_DEF_DATA   def_data;             /* defined data               */
} TP_LOAD_INFO;
```

```
typedef struct tp_load_info_def_data
{
  unsigned char           description[32];       /* Description               */
  unsigned char           reserv1[16];          /* reserved                   */
  unsigned char           user_id[64];          /* User ID                    */
  unsigned char           group_id[64];         /* Group ID                   */
  unsigned shor           timeout;              /* Timeout value              */
  unsigned char           type;                 /* TP type                    */
  unsigned char           reserv2;              /* reserved                   */
  AP_UINT16               reserv3;              /* reserved                   */
  AP_UINT16               ltv_length;           /* Length of LTV data         */
} TP_LOAD_INFO_DEF_DATA;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
照会中の TP_LOAD_INFO

buf_ptr
CS Linux が、要求された情報を戻すために使用するデータバッファへのポインタ。

buf_size
提供されるデータ・バッファーのサイズ。

num_entries
データが戻される必要がある追加のデータ制御ブロックの最大数。 特定の範囲ではなく、特定のリソースのデータを要求するには、 1 という値を指定します。 できるだけ多くのエントリーを戻すには、ゼロを指定します。この場合、 CS Linux は、指定されたデータ・バッファーに収容できる最大数のエントリーを戻します。

list_options
CS Linux がデータの戻りを開始する TP のリスト内の位置。 可能な値は次のとおりです

リストの最初のリスト （ _R ）
リストの最初の項目から開始します。

リストを含む （ 包括的 ）
TP 名と LU 別名の組み合わせによって指定されたエントリーから開始します。

次への AP_LIST_FROM_NEXT
TP 名と LU 別名の組み合わせによって指定されたエントリーの直後のエントリーから開始します。

リストの順序とアプリケーションでの特定のエントリーの取得方法について詳しくは、 34 ページの『List options for QUERY_* Verbs』を参照してください。

Tp_name
照会する TP 名。 これは 64 バイトの EBCDIC ストリングで、名前の長さが 64 文字より短い場合は、右側にスペースが埋め込まれます。 すべての TP 上で一致するすべての 2 進ゼロを指定します。 *list_options* を **リストの最初のリスト** （ _R ）に設定すると、この値は無視されます。

lu_alias
照会する LU 別名。 これは 8 バイトの ASCII ストリングで、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。 すべての LU で一致させるには、すべて 2 進ゼロを指定

このパラメーターは、 TP が APPC アプリケーションである場合にのみ使用できます。 TP が CPI-C アプリケーションである場合には、このパラメーターは予約されます。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

buf_size
Length of the information returned in the supplied buffer.

total_buf_size
Returned value indicating the size of buffer that would have been required to return all the list information requested. A value greater than *buf_size* indicates that not all the available entries were returned.

num_entries
Number of entries returned in the data buffer.

total_num_entries
Total number of entries available. A value greater than *num_entries* indicates that not all the available entries were returned.

Each entry in the data buffer consists of the following parameters:

tp_load_info.overlay_size
The size of this overlay, including the LTV data. This size includes padding to ensure that the next overlay falls on a properly aligned memory location.

When your application needs to go through the returned buffer to find each `tp_load_info` structure in turn, it must use this value to move to the correct offset for the next data structure, and must not use the C `sizeof()` operator. This is because the size of the returned overlay may increase in future releases of CS Linux; using the returned overlay size ensures that your application will continue to work with future releases.

tp_load_info.tp_name
TP name of the TP load info entry. This is a 64-byte EBCDIC string, padded on the right with spaces if the name is shorter than 64 characters.

tp_load_info.lu_alias
The LU alias of the TP load info entry. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes.

This parameter is used only if the TP is an APPC application; it is reserved if the TP is a CPI-C application.

def_data.description
Description of the TP load info.

def_data.user_id
User ID required to access and run the TP.

def_data.group_id
Group ID required to access and run the TP.

def_data.timeout
Timeout in seconds after the TP is loaded.

def_data.type
Indicates the TP type. Possible values are:

`AP_TP_TYPE_QUEUED`

`AP_TP_TYPE_QUEUED_BROADCAST`

`AP_TP_TYPE_NON_QUEUED`

def_data.ltv_length
Length of the LTV data buffer appended to this structure.

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
`AP_PARAMETER_CHECK`

secondary_rc
Possible values are:

AP_INVALID_TP_NAME
The *tp_name* parameter did not match the name of a defined TP.

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

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with `AP_PARAMETER_CHECK`, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

QUERY_TRACE_FILE

This verb returns information about the files that CS Linux uses to record trace data.

This verb must be issued to a running node.

VCB structure

```
typedef struct query_trace_file
{
   AP_UINT16        opcode;                  /* verb operation code       */
   unsigned char    reserv2;                 /* reserved                  */
   unsigned char    format;                  /* reserved                  */
   AP_UINT16        primary_rc;              /* primary return code       */
   AP_UINT32        secondary_rc;            /* secondary return code     */
   unsigned char    trace_file_type;         /* type of trace file        */
   unsigned char    dual_files;              /* dual trace files          */
   AP_UINT32        trace_file_size;         /* trace file size           */
   unsigned char    reserv3[4];              /* reserved                  */
   unsigned char    file_name[81];           /* file name                 */
   unsigned char    file_name_2[81];         /* second file name          */
} QUERY_TRACE_FILE;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
照会トレース・ファイル

trace_file_type
トレース・ファイルのタイプ。 可能な値は次のとおりです

追加 CS_TRACE
ファイルには、指定されたコンピューターと他のノード (SET_CS_TRACE verb によって活動化される) との間で CS Linux LAN 間で転送されるデータのトレースが含まれます。

TN_SERVER_TRACE のトレース
ファイルに、 CS Linux TN サーバー・コンポーネントのトレースが含まれています。

AP_IPS_TRACE トレース
ファイルには、指定されたノード (SET_TRACE_TYPE または ADD_DLC_TRACE verb によって活動化される) のカーネル・コンポーネントに対するトレースが含まれています。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

secondary_rc
Not used.

dual_files
Specifies whether tracing is to one file or to two files. Possible values are:

AP_YES
Tracing is to 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 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*.

AP_NO
Tracing is to one file.

trace_file_size
The maximum size of the trace file. If *dual_files* is set to AP_YES, tracing will switch between the two files when the current file reaches this size. If *dual_files* is set to AP_NO, this parameter is ignored; the file size is not limited.

file_name
Name of the trace file, or of the first trace file if *dual_files* is set to AP_YES. This parameter is an ASCII string of 1-80 characters, followed by a NULL character (binary zero).

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 is either a full 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 AP_YES. This parameter is an ASCII string of 1-80 characters, followed by a NULL character (binary zero).

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 is either a full path (starting with a / character) or the path relative to the default directory.

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc

AP_INVALID_FILE_TYPE
The *trace_file_type* parameter was not set to a valid value.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

QUERY_TRACE_TYPE

This verb returns information about the current tracing options for CS Linux kernel components. For more information about tracing options, see the *IBM Communications Server for Data Center Deployment on Linux Administration Guide*.

This verb does not return information about DLC line tracing. To do this, use the QUERY_DLC_TRACE verb.

This verb must be issued to a running node.

VCB structure

```
typedef struct query_trace_type
{
   AP_UINT16        opcode;               /* verb operation code            */
   unsigned char    reserv2;              /* reserved                       */
   unsigned char    format;               /* reserved                       */
   AP_UINT16        primary_rc;           /* primary return code            */
   AP_UINT32        secondary_rc;         /* secondary return code          */
   AP_UINT16        trace_flags;          /* trace flags                    */
   AP_UINT32        truncation_length;    /* truncate each msg to this size */
   AP_UINT16        internal_level;       /* reserved                       */
   AP_UINT32        api_flags;            /* reserved                       */
} QUERY_TRACE_TYPE;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
照会のトレース・タイプ

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

secondary_rc
Not used.

trace_flags
The types of tracing currently active. For more information about these trace types, see “SET_TRACE_TYPE” on page 582.

If no tracing is active, or if tracing of all types is active, this is one of the following values:

AP_NO_TRACE
No tracing.

AP_ALL_TRACE
Tracing of all types.

If tracing is being used on specific interfaces, this parameter is set to one or more values from the list below, combined using a logical OR operation.

AP_APPC_MSG
APPC messages

AP_FM_MSG
FM messages

AP_LUA_MSG
LUA messages

AP_NOF_MSG
NOF messages

AP_MS_MSG
MS messages

AP_GSNA_MSG
Generic SNA messages

AP_PV_MSG
(not used in this version of CS Linux)

AP_LLC2_MSG
LLC2 messages

AP_LLI_MSG
LLI messages

AP_MAC_MSG
MAC messages

AP_SDLC_MSG
SDLC messages

AP_NLI_MSG
NLI messages

AP_IPDL_MSG
Enterprise Extender (HPR/IP) messages

AP_DLC_MSG
Node to DLC messages

AP_NODE_MSG
Node messages

AP_SLIM_MSG
Messages sent between controller and backup servers in a client/server system

AP_DATAGRAM
Datagram messages

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, CS Linux writes only the start of the message to the trace file, and discards the data beyond *truncation_length*. This allows you to record the most important information for each message but avoid filling up the file with long messages. A value of zero indicates that trace messages are not truncated.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

QUERY_USERID_PASSWORD

QUERY_USERID_PASSWORD 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. It can be used to obtain information about a specific user ID / password pair or about multiple pairs, depending on the options used.

VCB structure

```
typedef struct query_userid_password
{
   AP_UINT16         opcode;                 /* Verb operation code         */
   unsigned char     reserv2;                /* reserved                    */
   unsigned char     format;                 /* reserved                    */
   AP_UINT16         primary_rc;             /* Primary return code         */
   AP_UINT32         secondary_rc;           /* Secondary return code       */
   unsigned char     *buf_ptr;               /* pointer to buffer           */
   AP_UINT32         buf_size;               /* buffer size                 */
   AP_UINT32         total_buf_size;         /* total buffer size required  */
   AP_UINT16         num_entries;            /* number of entries           */
   AP_UINT16         total_num_entries;      /* total number of entries     */
   unsigned char     list_options;           /* listing options             */
   unsigned char     reserv3;                /* reserved                    */
   unsigned char     user_id[10];            /* user ID                     */
} QUERY_USERID_PASSWORD;
```

```
typedef struct userid_info
{
   AP_UINT16              overlay_size;      /* size of returned entry      */
   unsigned char          user_id[10];       /* user ID                     */
   USERID_PASSWORD_CHARS  password_chars;    /* password characteristics    */
} USERID_INFO;
```

```
typedef struct userid_password_chars
{
   unsigned char          description[32];    /* resource description       */
   unsigned char          reserv2[16];        /* reserved                   */
   AP_UINT16              profile_count;      /* number of profiles         */
   AP_UINT16              reserv1;            /* reserved                   */
   unsigned char          password[10];       /* password                   */
   unsigned char          profiles[10][10];   /* profiles                   */
} USERID_PASSWORD_CHARS;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_QUERY_USERID_PASSWORD

buf_ptr
A pointer to a data buffer that CS Linux will use to return the requested information.

buf_size
Size of the supplied data buffer.

num_entries
Maximum number of user ID / password pairs for which data should be returned. To request a specific entry rather than a range, specify the value 1. To return as many entries as possible, specify zero; in this case, CS Linux will return the maximum number of entries that can be accommodated in the supplied data buffer.

list_options
The position in the list from which CS Linux should begin to return data. Specify one of the following values:

AP_FIRST_IN_LIST
Start at the first entry in the list.

AP_LIST_INCLUSIVE
Start at the entry specified by the *user_id* parameter.

AP_LIST_FROM_NEXT
Start at the entry immediately following the entry specified by the *user_id* parameter.

For more information about how the list is ordered and how the application can obtain specific entries from it, see “List options for QUERY_* Verbs” on page 34.

user_id
User ID. This is a 10-byte type-AE EBCDIC string, padded on the right with spaces if the name is shorter than 10 characters. The user ID is ignored if *list_options* is set to AP_FIRST_IN_LIST.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

buf_size
提供されたバッファーに戻された情報の長さ。

total_buf_size
要求されたすべてのリスト情報を戻すために必要となるバッファーのサイズを示す戻り値。 *buf_size* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

num_entries
データ・バッファーに戻されたエントリーの数。

total_num_entries
使用可能な項目の合計数。 *num_entries* より大きい値は、使用可能なすべてのエントリーが戻されなかったことを示します。

データ・バッファー内の各項目は、以下のパラメーターで構成されます。

***userid_info.*オーバーレイ・サイズ**
戻された **ユーザー ID 情報** 構造体のサイズ。すなわち、データ・バッファー内の次のエントリーの先頭までのオフセット。

アプリケーションが戻されたバッファーを調べて、各 **ユーザー ID 情報** 構造体を順番に検出する必要がある場合は、この値を使用して次のデータ構造の正しいオフセットに移動しなければなりませ

ん。また、 C `sizeof()` 演算子を使用してはなりません。 これは、返されるオーバーレイのサイズが、 CS Linux の将来のリリースで増加する可能性があるためです。戻されたオーバーレイ・サイズを使用すると、アプリケーションが今後のリリースで引き続き動作することが保証されます。

userid_info.user_id
ユーザー ID。 これは 10 バイトのタイプ AE の EBCDIC 文字ストリングで、右側に EBCDIC のスペースが埋め込まれます。

userid_info.password_chars.description
ユーザー ID とパスワードの定義に指定されている、ユーザー ID とパスワードを記述するヌル終了のテキスト・ストリング。

userid_info.password_chars.profile_count
このユーザーに定義されているプロファイルの数。

userid_info.password_chars.password
DEFINE_USERID_PASSWORD verb で指定されたユーザーのパスワードの暗号化されたバージョン。これは 10 バイトのタイプ AE の EBCDIC 文字ストリングで、右側に EBCDIC のスペースが埋め込まれます。

userid_info.password_chars.profiles
ユーザーに関連付けられたプロファイル。 これらはそれぞれ 10 バイトのタイプ AE の EBCDIC 文字ストリングで、右側に EBCDIC のスペースが埋め込まれます。

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

ファイル ID のユーザー ID
list_options パラメーターが **リストを含む** (**包括的**) に設定され、指定されたユーザー ID から始まるすべての項目がリストされましたが、 ユーザー *ID* パラメーターが無効でした。

ファイルの追加リスト・ オプション
list_options パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

登録受信側のシンク

REGISTER_INDICATION_SINK は、 NOF アプリケーションをレジスターに登録して、特定のタイプの指示を受信します。 CS Linux NOF 指示の詳細については、 603 ページの『第 4 章 NOF Indications』を参照してください。 アプリケーションは、 オペコード パラメーターによって必要とされる指示のタイプを指定します。アプリケーションは、複数の指標タイプを受け入れるために複数回登録することができます。 アプリケーションが (例えば、アプリケーションのターゲット・ノードの構成の変更や DLC の状況の変更など) 、アプリケーションが指示を要求したイベントが発生するたびに、 CS Linux は適切な指示メッセージをアプリケーションに送信します。

NOF_STATUS_INDICATION (ターゲット・ノードまたはファイルの状況の変更を示す) は、どのタイプの標識にも登録されているアプリケーションに戻されることがあります。 詳しくは、 636 ページの『NOF_STATUS_INDICATION』を参照してください。

この verb は、常に非同期 NOF API エントリー・ポイントを使用して発行する必要があります。コールバック・ルーチン (NOF API エントリー・ポイントの詳細については、 21 ページの『Asynchronous entry

point: nof_async』 を参照してください)。 CS Linux は、このコールバック・ルーチンを使用して、要求された指示をアプリケーションに戻します。

この verb は、以下のように、必要とされる指示のタイプに応じて、異なるターゲットに対して発行されることがあります。

- SNA ネットワーク・ファイル指示を登録するには、ターゲットが **スネネット** ファイル。 でなければなりません。
- サーバー指示を登録するには、ターゲットは必要ありません。アプリケーションは、ヌルのターゲット・ハンドルを指定する必要があります。
- ドメイン・リソースに関連する構成指示を登録するには、ターゲットがドメイン構成ファイルでなければなりません。
- ノード・リソースに関連する構成指示を登録するか、またはその他の指示を登録するには、ターゲットは、 CS Linux ソフトウェアが実行されているコンピューター上の実行中のノードまたは非アクティブ・ノードのいずれかになります。

VCB structure

```
typedef struct register_indication_sink
{
   AP_UINT16        opcode;               /* verb operation code             */
   unsigned char    reserv2;              /* reserved                        */
   unsigned char    format;               /* reserved                        */
   AP_UINT16        primary_rc;           /* primary return code             */
   AP_UINT32        secondary_rc;         /* secondary return code           */
   AP_UINT32        proc_id;              /* reserved                        */
   AP_UINT16        queue_id;             /* reserved                        */
   AP_UINT16        indication_opcode;    /* opcode of indication to be sunk */
} REGISTER_INDICATION_SINK;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_REGISTER_INDICATION_SINK

indication_opcode
The *opcode* parameter of the indication to be returned. CS Linux will send this indication to the application's callback routine every time the indication is generated.

To receive configuration indications, specify the value AP_CONFIG_INDICATION. If the target handle specified on the REGISTER_INDICATION_SINK verb identifies the domain configuration file, this value requests an indication each time the file is updated; if the target handle identifies a node, this value requests an indication each time the node's configuration is updated.

To receive SNA network file indications, issue the verb using a target handle that identifies the sna.net file, and specify the value AP_SNA_NET_INDICATION. This value requests an indication each time the file is updated.

For all other indications, specify the *opcode* value for the required indication. For more information, see the descriptions of individual indications in Chapter 4, “NOF Indications,” on page 603.

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

secondary_rc
未使用。

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_OP_CODE
One of the following has occurred:

- The *indication_opcode* parameter did not match the *opcode* of any of the CS Linux NOF API indications.
- The *indication_opcode* parameter specified an indication type that does not apply to the specified target. If the target handle identifies the domain configuration file, only configuration indications are valid; if the target handle identifies the `sna.net` file, only SNA network file indications are valid; and if the target handle specifies a running node, all indications except SNA network file indications are valid.

AP_DYNAMIC_LOAD_ALREADY_REGD
The *indication_opcode* parameter was set to a reserved value.

AP_SYNC_NOT_ALLOWED
The application issued REGISTER_INDICATION_SINK using the synchronous NOF entry point. This verb must use the asynchronous entry point.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: function not supported

If the verb does not execute successfully because the local node does not support the function associated with the specified indication, CS Linux returns the following parameters:

primary_rc

AP_FUNCTION_NOT_SUPPORTED
The local node does not support the specified indication. For details of the support required for each indication, see the description of each indication in Chapter 4, “NOF Indications,” on page 603.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

REMOVE_DLC_TRACE

This verb removes DLC line tracing that was previously specified using ADD_DLC_TRACE. It 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.

VCB structure

```
typedef struct remove_dlc_trace
{
   AP_UINT16         opcode;                 /* verb operation code        */
   unsigned char     reserv2;                /* reserved                   */
   unsigned char     format;                 /* reserved                   */
   AP_UINT16         primary_rc;             /* primary return code        */
   AP_UINT32         secondary_rc;           /* secondary return code      */
```

```
   DLC_TRACE_FILTER  filter;                  /* resource to stop tracing   */
} REMOVE_DLC_TRACE;
```

```
typedef struct dlc_trace_filter
{
   unsigned char     resource_type;           /* type of resource           */
   unsigned char     resource_name[8];        /* name of resource           */
   SNA_LFSID         lfsid;                   /* session identifier         */
   unsigned char     message_type;            /* type of messages           */
} DLC_TRACE_FILTER;
```

```
typedef struct sna_lfsid
{
      union
      {
          AP_UINT16            session_id;
          struct
          {
              unsigned char      sidh;
              unsigned char      sidl;
          } s;
      } uu;
      AP_UINT16       odai;
} SNA_LFSID;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_REMOVE_DLC_TRACE

resource_type
The resource type of the trace entry to remove or modify. Possible values are:

AP_ALL_DLC_TRACES
Remove all DLC tracing options, so that no resources are traced. If this option is specified, the remaining parameters on this verb *(resource_name* through *message_type*) are reserved.

AP_ALL_RESOURCES
Remove or modify the tracing options used for tracing all DLCs, ports, and LSs; resources for which DLC_TRACE entries are explicitly defined will continue to be traced.

AP_DLC
Remove or modify tracing for the DLC named in *resource_name*, and for all ports and LSs that use this DLC.

AP_PORT
Remove or modify tracing for the port named in *resource_name*, and for all LSs that use this port.

AP_LS
Remove or modify tracing for the LS named in *resource_name*.

AP_RTP
Remove or modify tracing for the RTP (rapid transport protocol) connection named in *resource_name*.

AP_PORT_DEFINED_LS
Modify tracing for the port named in *resource_name* and its defined LSs.

AP_PORT_IMPLICIT_LS
Modify tracing for the port named in *resource_name* and its implicit LSs.

resource_name
The name of the DLC, port, LS, or RTP connection for which tracing is being removed or modified. This parameter is reserved if *resource_type* is set to AP_ALL_DLC_TRACES or AP_ALL_RESOURCES.

lfsid
The Local Form Session Identifier for a session on the specified LS. This is only valid for *resource_type* AP_LS, and indicates that only messages on this session are to be removed. The structure contains

the following three values, which are returned in the SESSION_STATS section of a QUERY_SESSION verb:

lfsid.uu.s.sidh
Session ID high byte.

lfsid.uu.s.sidl
Session ID low byte.

lfsid.odai
Origin Destination Assignor Indicator.

message_type
The type of messages to trace for the specified resource or session. Set this parameter to AP_TRACE_ALL to remove all messages, or specify one or more of the following values (combined using a logical OR):

AP_TRACE_XID
XID messages

AP_TRACE_SC
Session Control RUs

AP_TRACE_DFC
Data Flow Control RUs

AP_TRACE_FMD
FMD messages

AP_TRACE_SEGS
Non-BBIU segments that do not contain an RH

AP_TRACE_CTL
Messages other then MUs and XIDs

AP_TRACE_NLP
(this message type is currently not used)

AP_TRACE_NC
(this message type is currently not used)

For tracing on an RTP connection, the values AP_TRACE_XID, AP_TRACE_NLP, and AP_TRACE_CTL are ignored.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

secondary_rc
Not used.

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のいずれかを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

ソース・リソース・タイプのタイプ
リソース・タイプ パラメーターに、無効な値が指定されました。

メッセージ・タイプの追加メッセージ・タイプ
メッセージ・タイプ パラメーターに、無効な値が指定されました。

ファイル名の変更ができない
リソース名 で指定された DLC には、トレース・オプションが設定されていません

ポートフォリオ・ポート名
リソース名 で指定されたポートには、トレース・オプション・セットがありません。

無効な LS_INVALID_LS_NAME
リソース名 の名前の LS には、トレース・オプションが設定されていません。

追加情報 _RTP_CONNECTION
リソース名 パラメーターで指定された RTP 接続には、トレース・オプションが設定されていません。

指定された LF_INVALID_LFSID_指定
リソース名 で指定された LS には、指定された LFSID に対してトレース・オプションが設定されていません。

入力ファイル・タイプの追加タイプ
メッセージ・タイプ パラメーターは、指定されたリソースについて現在トレースされていないメッセージ・タイプを指定しました。

追加の再リソースが定義されていない
リソース・タイプ パラメーターが **リソースの追加** （ _R ）に設定されていましたが、すべてのリソースのトレース・オプションに対して DLC_TRACE 項目が定義されていません。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、**パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

セッション限度のリセット

RESET_SESSION_LIMIT verb は、特定の LU - LU モードの組み合わせのセッション限度をリセットするために CS Linux を要求します。 この verb を処理した結果、セッションが非活動化される場合があります

VCB 構造体

```
typedef struct reset_session_limit
{
   AP_UINT16       opcode;                /* verb operation code            */
   unsigned char   reserv2;               /* reserved                       */
   unsigned char   format;                /* reserved                       */
   AP_UINT16       primary_rc;            /* primary return code            */
   AP_UINT32       secondary_rc;          /* secondary return code          */
   unsigned char   lu_name[8];            /* local LU name                  */
   unsigned char   lu_alias[8];           /* local LU alias                 */
   unsigned char   plu_alias[8];          /* partner LU alias               */
   unsigned char   fqplu_name[17];        /* fully qualified partner LU name*/
   unsigned char   reserv3;               /* reserved                       */
   unsigned char   mode_name[8];          /* mode name                      */
   unsigned char   mode_name_select;      /* select mode name               */
   unsigned char   set_negotiable;        /* set max negotiable limit to    */
                                          /*  zero?                         */
   unsigned char   reserv4[8];            /* reserved                       */
   unsigned char   responsible;           /* who is responsible for         */
                                          /*  deactivation                  */
   unsigned char   drain_source;          /* drain source                   */
   unsigned char   drain_target;          /* drain target                   */
   unsigned char   force;                 /* force                          */
   AP_UINT32       sense_data;            /* sense data                     */
} RESET_SESSION_LIMIT;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
追加リセット・セッション限度

lu_name
CS Linux に定義されている、ローカル LU の LU 名。 これは 8 バイトのタイプ A の EBCDIC ストリングで、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。 LU が LU 名の代わりに LU 別名で定義されていることを示すには、このパラメーターを 8 進ゼロに設定します。

lu_alias
CS Linux に定義されている、ローカル LU の LU 別名。 これは 8 バイトの ASCII ストリングで、ローカルで表示可能な文字を使用し、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。 *lu_name* がゼロに設定されている場合にのみ使用されます。

CP (デフォルト LU) に関連した LU を示すためには、 *lu_name* と *lu_alias* の両方を 2 進ゼロに設定します。

plu_alias
パートナー LU の LU 別名。

これは 8 バイトの ASCII ストリングで、ローカルで表示可能な文字を使用し、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。 パートナー LU が LU 別名の代わりに完全修飾 LU 名によって定義されていることを示すには、このパラメーターを 8 桁の 2 進ゼロに設定します。

fqplu_name
CS Linux に対して定義されている、パートナー LU の完全修飾 LU 名。 このパラメーターは、*plu_alias* フィールドがゼロに設定されている場合のみ使用されます。 *plu_alias* が指定されている場合は無視されます。

この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

モード名
セッション限度をリセットするモードの名前。 *mode_name_select* を **すべて追加**に設定すると、このパラメーターは無視されます。

これは 8 バイトの英数字のタイプ A の EBCDIC ストリング (文字で始まる) で、名前が 8 バイトより短い場合は、右側に EBCDIC のスペースが埋め込まれます。

mode_name_select
セッション限度を単一の指定モードでリセットするか、ローカル LU とパートナー LU の間のすべてのモードにリセットするかを選択します。 可能な値は次のとおりです

1 つの ID
モード名によって指定されたモードでセッション限度をリセットします。

すべて追加
すべてのモードのセッション限度をリセットする。

セット折衝可能
この LU - LU モードの組み合わせの最大折衝可能セッション限度をゼロにリセットする必要があるかどうかを指定します。 (現行の制限は、モードに指定された制限、または 初期セッション限度 または 変更セッション限度によって変更された可能性があります)。 可能な値は次のとおりです

類人猿
この LU - LU モードの組み合わせの最大折衝可能セッション限度をゼロにリセットして、INITIALIZE_SESSION_LIMIT によってセッションが変更されるまでセッションを活動化できないようにします。

アプ・ノー
交渉可能な最大セッション限度を未変更のままにします

責任者
セッション限度がリセットされた後に、ソース (ローカル) またはターゲット (パートナー) LU がセッションの非活動化を担当するかどうかを示します。 可能な値は次のとおりです

ソース・ソース
ローカル LU は、セッションの非活動化を担当します。

ターゲット・ターゲット
パートナー LU は、セッションの非活動化を担当します。

ソースのドレーン
セッションを非活動化する前に、ソース LU が待機セッション要求を満たすかどうかを指定 可能な値は次のとおりです

類人猿
待機中のセッション要求が満たされた。

アプ・ノー
待機セッション要求が満たされていません。

ターゲットのドレーン
セッションを非活動化する前に、ターゲット LU が待機セッション要求を満たすかどうかを指定 可能な値は次のとおりです

類人猿
待機中のセッション要求が満たされた。

アプ・ノー
待機セッション要求が満たされていません。

力
CNOS 折衝が失敗した場合でもセッション限度をゼロに設定するかどうかを指定します。 可能な値は次のとおりです

類人猿
セッション限度はゼロに設定されます。

アプ・ノー
CNOS 折衝が失敗した場合、セッション限度はゼロに設定されません。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

secondary_rc
Possible values are:

AP_FORCED
The session limits were set to zero even though CNOS negotiation failed.

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

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

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

許可される最大値 （ _MAX__ ）
CS Linux 内部エラーが発生しました。

エイプ無効ルリエイリアス
lu_alias パラメーターが、定義されたローカル LU 別名と一致しませんでした。

ファイル名の変更
lu_name パラメーターが、定義されたローカル LU 名と一致しませんでした。

ファイル・パス名
モード名 パラメーターが定義済みのどのモード名とも一致しませんでした

ファイル名の変更
fqplu_name パラメーターが、定義済みのパートナー LU 名と一致しませんでした。

ファイル名の選択解除 （ _INVALID_MODE_NAME_SELECT
mode_name_select パラメーターが有効な値に設定されていませんでした。

ソースの追加が妥当について
ソースのドレーン パラメーターが有効な値に設定されていませんでした。

ファイルの表示を無効にする （ _R ）
ターゲットのドレーン パラメーターが有効な値に設定されていませんでした。

使用不可の値
力 パラメーターが有効な値に設定されていませんでした。

無責任のアプインバリデータ
責任者 パラメーターが有効な値に設定されていませんでした。

付加的なセットのネゴシエーションが可能
セット折衝可能 パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: state check

If the verb does not execute because of a state error, CS Linux returns the following parameters.

primary_rc
AP_STATE_CHECK

secondary_rc

AP_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.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_STATE_CHECK, which are common to all NOF verbs.

Returned parameters: session allocation error

If the verb does not execute because of a session allocation error, CS Linux returns the following parameters:

primary_rc
AP_ALLOCATION_ERROR

secondary_rc

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

sense_data
The SNA sense data associated with the allocation failure.

戻りパラメーター: CNOS 処理エラー

エラーのために verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc

非 CONV_FAILURE_NO_RETRY
アクションを必要とする条件 (構成の不一致やセッション・プロトコル・エラーなど) のため、セッション限度を変更できませんでした。 CS Linux ログ・ファイルでエラー条件についての情報を確認し、この verb を再試行する前に訂正してください。

primary_rc

パートナーが LU_LU_リジェクトされた
CS Linux がパートナーとのセッション限度の折衝に失敗したため、 verb は失敗しました。 ローカル LU とパートナー LU の両方で構成を確認してください。

secondary_rc

AP_CNOS_COMMAND_RACE_REJECT
指定されたモードが、セッションの活動化または非活動化のため、またはセッション限度の処理のために、別の管理プログラム (または CS Linux ソフトウェアによって内部的に) によってアクセスされていたために、 verb は失敗しました。 競合状態がクリアされるようにするために、アプリケーションは verb を再試行する必要があります。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

SET_BUFFER_アベイラビリティー

この verb は、 CS Linux が任意の一時点で使用できる STREAMS バッファーの量を指定します。 これにより、ノードは使用可能なバッファーを効率的に使用することができ、また、バッファーが Linux コンピューター上の他のプロセスで使用可能であることを確認することができます。

VCB structure

```
typedef struct set_buffer_availability
{
   AP_UINT16        opcode;              /* verb operation code             */
   unsigned char    reserv2;             /* reserved                        */
   unsigned char    format;              /* reserved                        */
   AP_UINT16        primary_rc;          /* primary return code             */
   AP_UINT32        secondary_rc;        /* secondary return code           */
   AP_UINT32        buf_avail;           /* maximum buffer space available  */
   unsigned char    reserv3[8];          /* reserved                        */
} SET_BUFFER_AVAILABILITY;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
追加セット・バッファーの可用性

buf_avail
使用可能な STREAMS バッファー・スペースの最大量 (バイト単位) 。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

secondary_rc
未使用。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

SET_CENTRAL_LOGGING

This verb specifies whether CS Linux log messages are sent to a central file from all servers, or to a separate file on each server. For more information, see “SET_LOG_FILE” on page 573.

This verb must be issued to the node that is currently acting as the central logger; for information about accessing this node, see “接続ノード” on page 53.

VCB structure

```
typedef struct set_central_logging
{
   AP_UINT16        opcode;                 /* verb operation code          */
   unsigned char    reserv2;                /* reserved                     */
   unsigned char    format;                 /* reserved                     */
   AP_UINT16        primary_rc;             /* primary return code          */
   AP_UINT32        secondary_rc;           /* secondary return code        */
   unsigned char    enabled;                /* is central logging enabled?  */
   unsigned char    reserv3[3];             /* reserved                     */
} SET_CENTRAL_LOGGING;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
AP_SET_CENTRAL_LOGGING

有効
中央ロギングを使用可能または使用不可にするかどうか 可能な値は次のとおりです

類人猿
セントラル・ロギングは使用可能です。 すべてのログ・メッセージは、中央ロガーとして現在活動しているノード上の単一ファイルに送信されます。

アプ・ノー
中央ロギングは使用不可です。 各サーバーからのログ・メッセージは、そのサーバー上のファイル (SET_LOG_FILE verb を使用して指定) に送信されます。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

secondary_rc
未使用。

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc

AP_NOT_CENTRAL_ロガー
この verb は、中央ロガーではないノードに対して発行されました。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

SET_CS_TRACE

This verb specifies tracing options for data sent between computers on the CS Linux LAN. For more information about tracing options, see the *IBM Communications Server for Data Center Deployment on Linux Administration Guide*.

This verb can be issued from a NOF application running on an AIX or Linux client. The NOF application must run with the userid `root`, or with a userid that is a member of the `sys` group (AIX) or `sna` group (Linux).

This verb must be issued to a running node, unless it is issued from a client.

VCB structure

```
typedef struct set_cs_trace
{
   AP_UINT16        opcode;            /* verb operation code               */
   unsigned char    reserv2;           /* reserved                          */
   unsigned char    format;            /* reserved                          */
   AP_UINT16        primary_rc;        /* primary return code               */
   AP_UINT32        secondary_rc;      /* secondary return code             */
   unsigned char    dest_sys[128];     /* node to which messages are traced  */
   unsigned char    reserv4[4];        /* reserved                          */
   AP_UINT16        trace_flags;       /* trace flags                       */
   AP_UINT16        trace_direction;   /* direction (send/rcv/both) to trace */
   unsigned char    reserv3[8];        /* reserved                          */
} SET_CS_TRACE;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
追加セット CS_TRACE

dest_sys
トレースを必要とするサーバー名。 これは ASCII ストリングで、名前が 128 文字より短い場合は、右側にスペースが埋め込まれます。

この verb が発行されたコンピューター (NOF API 呼び出しの *target_handle* パラメーターによって識別される) と LAN 上の他の 1 つのサーバーとの間に流れるメッセージ・フローのトレースを管理するには、他のサーバーの名前をここに指定します。 LAN 上の他のコンピューターとの間でやり取りされるメッセージのトレースは変更されません。特に、 2 つの SET_CS_TRACE verb を発行して、同じターゲット・コンピューターと 2 つの異なる宛先サーバーの間でトレースを活動化することができます。

サーバー名に . (ピリオド) 文字が含まれている場合、 CS Linux は、サーバー名が完全修飾名であると見なします。それ以外の場合は、 DNS ルックアップを実行してサーバー名を判別します。

この verb が発行されたコンピューター (NOF API 呼び出しの *target_handle* パラメーターによって識別される) と、 LAN 上の他のすべてのサーバーとクライアントとの 間で流れているメッセージのトレースを管理するには、このパラメーターを 128 ASCII スペース文字に設定します。 この verb で指定するオプションは、特定のコンピューター (前の verb では *dest_sys* によって識別される) へのトレース用の以前の設定をオーバーライドします。

trace_flags
必要なトレースのタイプ。 すべてのトレースをオフにするか、すべてのタイプのトレースをオンにするには、以下のいずれかの値を指定します。

AP_NO_TRACE
トレースなし。

すべてのトレースの追加
すべてのタイプのトレース。

特定のメッセージ・タイプに対してトレースをアクティブにするには、以下のリストから 1 つ以上の値を選択し、論理 **それとも** 操作を使用して結合します。

アプ CS_ADMIN_MSG
クライアント/サーバー・トポロジーに関連する内部メッセージ

アプリケーション・データグラム (_C)
データグラム・メッセージ

付加 CS_DATA
データ・メッセージ

trace_direction
トレースが必要とされる方向を指定します。 *trace_flags* を AP_NO_TRACE に設定すると、このパラメーターは無視されます。 可能な値は次のとおりです

アプ CS_SEND
ターゲット・コンピューターから *dest_sys* によって定義されたコンピューターに流れるメッセージをトレースします。

追加 CS_RECEIVE
dest_sys によって定義されたコンピューターからターゲット・コンピューターに流れるメッセージをトレースします。

両方の CS_CS_両方
両方向に流れるメッセージをトレースします。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

secondary_rc
未使用。

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_NAME_NOT_FOUND
The server specified by the *dest_sys* parameter did not exist or was not started.

AP_LOCAL_SYSTEM
The server specified by the *dest_sys* parameter is the same as the target node to which this verb was issued.

AP_INVALID_TRC_DIRECTION
The *trace_direction* parameter was not set to a valid value.

AP_INVALID_TARGET
The verb was issued on a standalone server. This verb can only be issued on a client/server system.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

SET_GLOBAL_LOG_TYPE

This verb specifies the types of information that CS Linux records in log files. It specifies default values that are used on all servers; SET_LOG_TYPE can then be used to override these defaults on a particular server. For more information about log files, see “SET_LOG_FILE” on page 573.

CS Linux logs messages for the following types of event:

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

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

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

Problem and exception messages are logged to the error log file; audit messages are logged to the audit log file. Problem messages are always logged and cannot be disabled, but you can specify whether to log each of the other two message types. For each of the two files (audit and error), you can specify whether to use succinct logging (including only the text of the message and a summary of the message source) or full logging (including full details of the message source, cause, and any action required).

This verb must be issued to the node currently acting as the central logger; for more information, see “接続ノード” on page 53.

VCB structure

```
typedef struct set_global_log_type
{
   AP_UINT16       opcode;             /* verb operation code               */
   unsigned char   reserv2;            /* reserved                          */
   unsigned char   format;             /* reserved                          */
   AP_UINT16       primary_rc;         /* primary return code               */
   AP_UINT32       secondary_rc;       /* secondary return code             */
   unsigned char   audit;              /* audit logging on or off           */
   unsigned char   exception;          /* exception logging on or off       */
   unsigned char   succinct_audits;    /* use succinct logging in audit file?*/
   unsigned char   succinct_errors;    /* use succinct logging in error file?*/
   unsigned char   reserv3[4];         /* reserved                          */
} SET_GLOBAL_LOG_TYPE;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
アプリケーション・ログ・ログ・タイプの追加

監査
監査メッセージを記録するかどうかを指定 可能な値は次のとおりです

類人猿
監査メッセージが記録される。

アプ・ノー
監査メッセージは記録されません。

無変更の付加
監査ログを既存の定義から変更しないでください。(CS Linux ソフトウェアが開始される初期のデフォルトは、監査メッセージが記録されていないことです。)

例外
例外メッセージを記録するかどうかを指定 可能な値は次のとおりです

類人猿
例外メッセージが記録される。

アプ・ノー
例外メッセージは記録されません。

無変更の付加
例外ロギングは、既存の定義から変更しないでください。(CS Linux ソフトウェアが開始される初期のデフォルトは、例外メッセージが記録されることです。)

正常に監査されます
監査ログ・ファイルで簡略ロギングを使用するか、フル・ロギングを使用するかを指定します。 可能な値は次のとおりです

類人猿
簡潔ロギング: ログ・ファイル内の各メッセージには、メッセージ・ヘッダー情報 (メッセージ番号、ログ・タイプ、システム名など) とメッセージ・テキスト・ストリングおよびパラメーターの要約が含まれています。 ログの原因、および必要なアクションの詳細を取得するには、 snahelp ユーティリティーを 使用できます。

アプ・ノー
フル・ロギング: ログ・ファイル内の各メッセージには、メッセージ・ヘッダー情報の完全なリスト、メッセージ・テキスト・ストリングとパラメーター、およびログの原因に関する追加情報と、必要なアクションが含まれています。

無変更の付加
前の SET_GLOBAL_LOG_TYPE verb でこのパラメーターに指定された値 (簡略ログまたはフル・ロギング) を使用します。 SET_GLOBAL_LOG_TYPE verb が発行される前の初期デフォルト値は、簡略ログを使用することです。

セントラル・ロギングを使用している場合、すべてのコンピューターからのメッセージの簡潔なロギングまたはフル・ログの選択は、中央ロガーとして機能するサーバー上でこのパラメーターの設定によって決まります。この設定は、 SET_GLOBAL_LOG_TYPE verb からのものであるか、またはデフォルトをオーバーライドするためにサーバーに 発行された SET_LOG_TYPE verb のいずれかになっている可能性があります。

正常性エラー
エラー・ログ・ファイルで簡略ロギングまたはフル・ロギングを使用するかどうかを指定します。これは、例外ログと問題ログの両方に適用されます。 許可される値とその意味は、 正常に監査されますパラメーターの場合と同じです。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

secondary_rc
未使用。

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_NOT_CENTRAL_LOGGER
The verb was issued to a node that is not the central logger.

AP_INVALID_SUCCINCT_SETTING
The *succinct_audits* or *succinct_errors* parameter was not set to a valid value.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

SET_KERNEL_MEMORY_LIMIT

This verb specifies a limit on the amount of kernel memory that CS Linux can use at any one time. This allows 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 CS Linux software (for more information, see the *IBM Communications Server for Data Center Deployment on Linux Administration Guide*). This verb overrides the limit, if any, specified when starting the CS Linux software.

VCB 構造体

```
typedef struct set_kernel_memory_limit
{
   AP_UINT16       opcode;           /* verb operation code                */
   unsigned char   reserv2;          /* reserved                           */
   unsigned char   format;           /* reserved                           */
   AP_UINT16       primary_rc;       /* primary return code                */
   AP_UINT32       secondary_rc;     /* secondary return code              */
   AP_UINT32       limit;            /* kernel memory limit, 0 => no limit  */
   unsigned char   reserv3[8];       /* Reserved                           */
} SET_KERNEL_MEMORY_LIMIT;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_SET_KERNEL_MEMORY_LIMIT

limit
The maximum amount of kernel memory that CS Linux should use at any time, in bytes. If a CS Linux component attempts to allocate kernel memory that would take the total amount of memory currently allocated to CS Linux components above this limit, the allocation attempt will fail.

To remove the limit set by a previous SET_KERNEL_MEMORY_LIMIT verb, specify zero.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

secondary_rc
Not used.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

SET_LOG_FILE

This verb manages a file that CS Linux uses to record log messages. It allows 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, CS 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 SET_CENTRAL_LOGGING, this verb 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 verb can be issued from a NOF application running on an AIX or Linux client. The NOF application must run with the userid `root`, or with a userid that is a member of the `sys` group (AIX) or `sna` group (Linux).

VCB 構造体

```
typedef struct set_log_file
{
   AP_UINT16       opcode;                 /* verb operation code          */
   unsigned char   reserv2;                /* reserved                     */
   unsigned char   format;                 /* reserved                     */
   AP_UINT16       primary_rc;             /* primary return code          */
   AP_UINT32       secondary_rc;           /* secondary return code        */
   unsigned char   log_file_type;          /* type of log file             */
   unsigned char   action;                 /* reset and/or backup existing */
   unsigned char   file_name[81];          /* file name                    */
   unsigned char   backup_file_name[81];   /* backup file                  */
   AP_UINT32       file_size;              /* log file size                */
   unsigned char   succinct;               /* reserved                     */
   unsigned char   reserv3[3];             /* reserved                     */
} SET_LOG_FILE;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_SET_LOG_FILE

log_file_type
The type of log file being managed. Possible values are:

AP_AUDIT_FILE
Audit log file (audit messages only).

AP_ERROR_FILE
Error log file (problem and exception messages).

AP_USAGE_FILE
Usage log file (information on current and peak usage of CS Linux resources).

To record both audit and error messages in the same file, issue two SET_LOG_FILE verbs for the same file name, specifying AP_AUDIT_FILE on one verb and AP_ERROR_FILE on the other.

action
The action to be taken on the log file. Specify one of the following values:

AP_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 verb completes successfully, all log messages of the type defined by *log_file_type* are written to the new log file. The log file that was used before this verb is issued, if any, is left unchanged.

AP_DELETE_FILE
Delete the contents of the current log file.

AP_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 relative to the application's working directory or a full path) on any computer to which this verb is issued.

This parameter is an ASCII string of 1-80 characters, followed by a NULL character (binary zero). To continue logging to the file specified on a previous SET_LOG_FILE verb, specify a null string.

backup_file_name
Name of the backup log file. When the log file reaches the size specified by *file_size* below, CS 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 above.

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 relative to the application's working directory or a full path) on any computer to which this verb is issued.

This parameter is an ASCII string of 1-80 characters, followed by a NULL character (binary zero). To continue using the backup file specified on a previous SET_LOG_FILE verb, specify a null string.

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, CS Linux copies the current contents of the log file to the backup log file and clears the log file. This means that the maximum amount of disk space taken up by log files is approximately twice *file_size*.

To continue using the file size specified on a previous SET_LOG_FILE verb, set this parameter to zero. The initial default value, before any SET_LOG_FILE verb has been issued, is 10,000,000 bytes. A value of zero indicates "continue using the existing file size" and not "no limit".

You may need to increase the size of the audit and error log files according to the size of the CS 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:

- Large numbers of clients or users (since 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 full logging instead of succinct logging.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

secondary_rc
Not used.

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

ファイル・ファイル・アクションのアクション
アクション パラメーターが有効な値に設定されていませんでした。

ファイル・ファイル・タイプの追加
ログ・ファイル・タイプ パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

ログ・タイプの設定

この verb は、 CS Linux レコードが特定のサーバー上のログ・ファイルに記録する情報のタイプを指定します。 これを使用して、 SET_GLOBAL_LOG_TYPE で指定されたデフォルト設定をオーバーライドするか、または指定変更を除去して、このサーバーがデフォルト設定値を使用できるようにすることができます。 ログ・ファイルについて詳しくは、 573 ページの『SET_LOG_FILE』を参照してください。

この verb は、 AIX または Linux クライアント上で実行される NOF アプリケーションから発行できます。 NOF アプリケーションは、ユーザー ID **根**で実行するか、または **システム** グループ (AIX) または **スナバ** グループ (Linux) のメンバーであるユーザー ID を使用して実行する必要があります。

問題
ユーザー (セッションの異常終了など) に認識できないようにシステムを機能低下させる異常イベント。

例外
システムを低下させる異常なイベントですが、即時にユーザー (リソース不足など) には認識されないか、システムを低下させないイベントが発生しますが、後の例外や問題の原因 (リモート・システムから予期しないメッセージを受信するなど) を示している場合があります。

監査
通常のイベント (セッションの開始など) 。

問題メッセージおよび例外メッセージは、エラー・ログ・ファイルに記録されます。監査メッセージは監査ログ・ファイルに記録されます。 問題メッセージは常にログに記録され、使用不可にすることはできませんが、他の 2 つのメッセージ・タイプをそれぞれログに記録するかどうか 2 つのファイル (監査およびエラー) のそれぞれについて、簡潔なロギングを使用するか (メッセージのテキストとメッセージ・ソースの要約のみを含む) 、またはフル・ロギング (メッセージ・ソースの完全な詳細、原因、および必要なアクションを含む) を指定することができます。

VCB 構造体

```
typedef struct set_log_type
{
```

```
   AP_UINT16       opcode;             /* verb operation code               */
   unsigned char   reserv2;            /* reserved                          */
   unsigned char   format;             /* reserved                          */
   AP_UINT16       primary_rc;         /* primary return code               */
   AP_UINT32       secondary_rc;       /* secondary return code             */
   unsigned char   override;           /* override global defaults?         */
   unsigned char   audit;              /* audit logging on or off           */
   unsigned char   exception;          /* exception logging on or off       */
   unsigned char   succinct_audits;    /* use succinct logging in audit file?*/
   unsigned char   succinct_errors;    /* use succinct logging in error file?*/
   unsigned char   reserv3[3];         /* reserved                          */
} SET_LOG_TYPE;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_SET_LOG_TYPE

override
Specifies whether this verb is being used to override the global log types specified on SET_GLOBAL_LOG_TYPE, or to revert to using these defaults. Possible values are:

AP_YES
Override the global log types. The log types to be used on this server are specified by the *audit* and *exception* parameters below, and the choice of succinct or full logging is specified by the *succinct_** parameters below.

AP_NO
Revert to using the global log types. The *audit*, *exception*, and *succinct_** parameters below are ignored.

audit
Specify whether audit messages are recorded on this server. Possible values are:

AP_YES
Audit messages are recorded.

AP_NO
Audit messages are not recorded.

AP_LEAVE_UNCHANGED
Leave audit logging unchanged from the existing definition.

exception
Specify whether exception messages are recorded on this server. Possible values are:

AP_YES
Exception messages are recorded.

AP_NO
Exception messages are not recorded.

AP_LEAVE_UNCHANGED
Leave exception logging unchanged from the existing definition.

succinct_audits
Specifies whether to use succinct logging or full logging in the audit log file on this server. Possible values are:

AP_YES
Succinct logging: 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 of the cause of the log and any action required, you can use the snahelp utility.

AP_NO
Full logging: each message in the log file includes a full listing of the message header information, the message text string and parameters, and additional information about the cause of the log and any action required.

AP_LEAVE_UNCHANGED
Leave succinct logging or full logging unchanged from the existing definition.

If you are using central logging, the choice of succinct or full logging for messages from all computers is determined by the setting of this parameter on the server acting as the central logger; this setting may either be from the SET_GLOBAL_LOG_TYPE verb, or from a SET_LOG_TYPE verb issued to that server to override the default.

succinct_errors
Specifies whether to use succinct logging or full logging in the error log file on this server; this applies to both exception logs and problem logs. The allowed values and their meanings are the same as for the *succinct_audits* parameter.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

secondary_rc
Not used.

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc

AP_INVALID_SUCCINCT_SETTING
The *succinct_audits* or *succinct_errors* parameter was not set to a valid value.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

SET_PROCESSING_MODE

This verb specifies how the NOF application interacts with the target node, configuration file, or SNA network data file: whether the application has read-only access or read/write access, and whether the application has exclusive access to the domain configuration file so that other applications cannot access it.

This verb applies only to NOF applications running on a server. For an application running on a client, the only processing mode available is read-only mode (the default), in which the application can issue QUERY_* verbs but cannot define, start or stop resources. The client application cannot use SET_PROCESSING_MODE to select any other mode.

The target node or file is specified by the *target_handle* parameter on the NOF API call; the application obtains this parameter from the verb CONNECT_NODE (for a node) or OPEN_FILE (for a file). For more information about the use of this parameter, see “NOF API entry points for AIX or Linux” on page 19.

This verb may be issued to the domain configuration file, to the `sna.net` file, or to a running node. The valid processing modes that can be set with this verb depend on the target type.

VCB structure

```
typedef struct set_processing_mode
{
   AP_UINT16        opcode;           /* verb operation code               */
   unsigned char    reserv2;          /* reserved                          */
   unsigned char    format;           /* reserved                          */
   AP_UINT16        primary_rc;       /* primary return code               */
   AP_UINT32        secondary_rc;     /* secondary return code             */
   unsigned char    mode;             /* new mode to be set for this handle */
   AP_UINT16        reserv1;          /* reserved                          */
} SET_PROCESSING_MODE;
```

Supplied parameters

opcode
AP_SET_PROCESSING_MODE

mode
Requested mode for this target handle. The mode cannot be changed while any previous verbs issued using this target handle are still outstanding. Possible values are:

AP_MODE_READ_ONLY
Read-only mode: the application will use only QUERY_* verbs, which do not modify the configuration. This option can be used with either a file or a node as the target.

AP_MODE_READ_WRITE
Read / write mode: the application may use any NOF API verbs. This option can be used with either a file or a node as the target.

AP_MODE_COMMIT
Commit mode: the application has exclusive read/write access to the target file, so that other applications cannot access it until this application releases it. This option can be used only with the domain configuration file as the target.

This mode is intended for issuing a series of connected verbs to a file (such as a series of DEFINE verbs for related components). The application should complete the series of verbs as quickly as possible and then reset its processing mode to one of the other options, in order to release the file so that other NOF API applications or CS Linux components can access it.

Note : To obtain read/write or commit access to the file, your NOF application must be running with a user ID that is a member of the SNA administrators group `sna` (or running as `root`). If the user ID is not a member of this group or `root`, the only valid processing mode is AP_MODE_READ_ONLY.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

secondary_rc
Not used.

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_PROC_MODE
The *mode* parameter was not set to a valid value.

AP_INVALID_TARGET_MODE
The *mode* parameter was not valid for the selected target.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: state check

If the verb does not execute because of a state check, CS Linux returns the following parameters.

primary_rc
AP_STATE_CHECK

secondary_rc
Possible values are:

AP_FILE_UNAVAILABLE
The application specified commit mode, but was unable to get exclusive access to the required configuration file. This may be because another application is accessing the file in commit mode.

AP_VERB_IN_PROGRESS
The processing mode for the specified target handle cannot be changed because a previous verb issued for this handle is still outstanding. All verbs for the target handle must be completed before attempting to change the processing mode.

AP_NOT_AUTHORIZED
The NOF application cannot obtain read/write access to the file because it is running on a client, or because it is running with a user ID that is not a member of the SNA administrators group sna. If the user ID is not a member of this group, the only valid processing mode is AP_MODE_READ_ONLY.

AP_NOT_CONTROLLER
The processing mode cannot be changed to AP_MODE_READ_WRITE or AP_MODE_COMMIT because the target handle specifies a file (either the domain configuration file or the SNA network data file) on a backup server that is no longer acting as the controller server. Changes to the running configuration file can be made only to the copy of this file on the controller (so that they will be distributed to other servers); other copies of the file can be accessed only in read-only mode. If the application needs to use read/write or commit mode, it should issue CLOSE_FILE for this target handle, and then reissue OPEN_FILE to access the file on the new controller server.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_STATE_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

SET_TN_SERVER_TRACE

This verb specifies tracing options for the CS Linux TN server component.

This verb must be issued to a running node.

VCB structure

```
typedef struct set_tn_server_trace
{
   AP_UINT16        opcode;                  /* verb operation code      */
   unsigned char    reserv2;                 /* reserved                 */
   unsigned char    format;                  /* reserved                 */
   AP_UINT16        primary_rc;              /* primary return code      */
   AP_UINT32        secondary_rc;            /* secondary return code    */
   AP_UINT16        trace_flags;             /* trace flags              */
```

```
   unsigned char    reserv3[6];                 /* reserved                 */
} SET_TN_SERVER_TRACE;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
アプリケーション・セットの追加サーバー・トレース

trace_flags
必要なトレースのタイプ。 すべてのトレースをオフにするか、すべてのタイプのトレースをオンにするには、以下のいずれかの値を指定します。

TN_SERVER_NO_TRACE
トレースなし。

TN_SERVER_ALL_TRACE の呼び出し
すべてのタイプのトレース。

特定のメッセージ・タイプに対してトレースをアクティブにするには、以下のリストから 1 つ以上の値を選択し、論理 **それとも** 操作を使用して結合します。

___ _SERVER_TRC_TCP
TCP/IP インターフェース ・トレース: TN サーバーと TN3270 クライアントの間のメッセージ

トランザクション・サーバーの TRC_FM
ノード・インターフェースのトレース : 内部制御メッセージ、および TN サーバーと TN3270 クライアントの間のメッセージ (内部フォーマット)

TN_SERVER_TRC_CFG を追加します。
構成メッセージ・トレース: TN サーバーの構成に関連するメッセージ

表サーバーのトランザクション・トランザクション （ _N ）
内部ノード・オペレーター機能 (NOF) のトレース: TN サーバーによって行われたトレース NOF 要求。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

secondary_rc
Not used.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

SET_TRACE_FILE

This verb specifies the name of a file that CS Linux uses to record trace data.

If you issue a second SET_TRACE_FILE verb specifying a new file for the same file type, all subsequent trace information will be written to the new file; the existing file is not removed, but further information will not be written to it. If you issue a second SET_TRACE_FILE verb for the same trace file, this resets the file (discarding trace information that was written to the file before the second verb).

This verb must be issued to a running node.

VCB structure

```
typedef struct set_trace_file
{
   AP_UINT16        opcode;                    /* verb operation code    */
   unsigned char    reserv2;                   /* reserved               */
   unsigned char    format;                    /* reserved               */
   AP_UINT16        primary_rc;                /* primary return code    */
   AP_UINT32        secondary_rc;              /* secondary return code  */
   unsigned char    trace_file_type;           /* type of trace file     */
   unsigned char    dual_files;                /* dual trace files       */
   AP_UINT32        trace_file_size;           /* trace file size        */
   unsigned char    reserv3[4];                /* reserved               */
   unsigned char    file_name[81];             /* file name              */
   unsigned char    file_name_2[81];           /* second file name       */
} SET_TRACE_FILE;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_SET_TRACE_FILE

trace_file_type
The type of trace file. Possible values are:

AP_CS_TRACE
File contains tracing on data transferred across the CS Linux LAN between the specified computer and other nodes (activated by the SET_CS_TRACE verb).

AP_TN_SERVER_TRACE
File contains tracing on the CS Linux TN server component.

AP_IPS_TRACE
File contains tracing on kernel components for the specified node (activated by the SET_TRACE_TYPE or ADD_DLC_TRACE verb).

dual_files
Specifies whether tracing is to one file or to two files. Possible values are:

AP_YES
Tracing is to 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 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*.

AP_NO
Tracing is to one file.

AP_LEAVE_UNCHANGED
Leave the *dual_files* setting unchanged from the existing definition. (The initial default, when the CS Linux software is started, is to use two files.)

trace_file_size
The maximum size of the trace file, in bytes. The initial default value, before any SET_TRACE_FILE verb has been issued, is 10,000,000 bytes. To continue using the existing file size definition, specify zero.

If *dual_files* is set to AP_YES, tracing will switch between the two files when the current file reaches this size. If *dual_files* is set to AP_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 CS Linux client/server network, to allow for the volume of trace information generated in larger systems. In particular, 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 of the first trace file if *dual_files* is set to AP_YES. To continue using the file name specified on a previous SET_TRACE_FILE verb, set this parameter to a null string.

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 relative to the application's working directory or a full path) on any computer to which this verb is issued.

This parameter is an ASCII string of 1-80 characters, followed by a NULL character (binary zero).

file_name_2
Name of the second trace file; this parameter is used only if *dual_files* is set to AP_YES. To continue using the file name specified on a previous *set_trace_file* verb, set this parameter to a null string.

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 relative to the application's working directory or a full path) on any computer to which this verb is issued.

This parameter is an ASCII string of 1-80 characters, followed by a NULL character (binary zero).

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

secondary_rc
Not used.

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

追加ファイル・ ファイル名
ファイル名 または ファイル名 *2* パラメーターが有効な Linux ファイル名に設定されていなかったか、または単一のトレース・ファイルから重複トレース・ファイルへの変更時に ファイル名 *2* が指定されませんでした。

ファイル・ ファイル・ タイプの追加
trace_file_type パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

SET_TRACE_TYPE

This verb specifies tracing options for CS Linux kernel components. You can use this verb 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, see the *IBM Communications Server for Data Center Deployment on Linux Administration Guide*.

To control DLC line tracing, use the ADD_DLC_TRACE verb. The truncation length specified on this verb also applies to DLC tracing, but the tracing options on this verb do not apply to DLC tracing.

This verb must be issued to a running node.

VCB 構造体

```
typedef struct set_trace_type
{
   AP_UINT16        opcode;               /* verb operation code            */
   unsigned char    reserv2;              /* reserved                       */
   unsigned char    format;               /* reserved                       */
   AP_UINT16        primary_rc;           /* primary return code            */
   AP_UINT32        secondary_rc;         /* secondary return code          */
   AP_UINT16        trace_flags;          /* trace flags                    */
   AP_UINT32        truncation_length;    /* truncate each msg to this size */
   unsigned char    init_flags;           /* TRUE if initializing flags     */
   unsigned char    set_flags;            /* TRUE if setting flags          */
                                          /*  FALSE if unsetting flags      */
   unsigned char    set_internal;         /* reserved                       */
   AP_UINT16        internal_level;       /* reserved                       */
   AP_UINT32        api_flags;            /* api trace flags                */
} SET_TRACE_TYPE;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_SET_TRACE_TYPE

trace_flags
The types of tracing required. To turn off all tracing, or to turn on tracing of all types, specify one of the following values:

AP_NO_TRACE
No tracing.

AP_ALL_TRACE
Tracing of all types.

To control tracing on specific interfaces, select one or more values from the list below, combined using a logical OR operation. For more information about these trace types, see “Trace types” on page 585.

If *init_flags* is set to AP_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 AP_NO, select the values corresponding to the interfaces where you want to change the state of tracing.

AP_APPC_MSG
APPC messages

AP_LUA_MSG
LUA messages

AP_NOF_MSG
NOF messages

AP_MS_MSG
MS messages

AP_LLC2_MSG
LLC2 messages

AP_LLI_MSG
LLI messages

AP_MAC_MSG
MAC messages

AP_SDLC_MSG
SDLC messages (note that this option also provides additional detail in SDLC line tracing)

AP_NLI_MSG
NLI messages

AP_IPDL_MSG
Enterprise Extender (HPR/IP) messages

AP_DLC_MSG
Node to DLC messages

AP_NODE_MSG
Node messages

AP_SLIM_MSG
Messages sent between controller and backup servers

AP_DATAGRAM
Datagram messages

truncation_length
Specify the maximum length, in bytes, of the information to be written to the trace file for each message. This value must be at least 256.

If a trace message is longer than the length specified in this parameter, CS Linux writes only the start of the message to the trace file, and discards the data beyond *truncation_length*. This allows you to record the most important information for each message but avoid filling up the file with long messages.

To specify no truncation (all the data from each message is written to the file), set this parameter to zero.

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:

AP_YES
Tracing is being initialized. The *trace_flags* parameter defines the required state of tracing at all interfaces.

AP_NO
Tracing is being changed. The *trace_flags* parameter defines the interfaces where tracing is being activated or deactivated; other interfaces will not be affected.

set_flags
If *init_flags* is set to AP_NO, this parameter specifies whether tracing is to be activated or deactivated at the requested interfaces. Possible values are:

AP_YES
Tracing is to be activated at the interfaces specified by the *trace_flags* parameter.

AP_NO
Tracing is to be deactivated at the interfaces specified by the *trace_flags* parameter.

If *init_flags* is set to AP_YES, this parameter is ignored.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

secondary_rc
Not used.

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc

AP_INVALID_TRUNC_LEN
The *truncation_length* parameter specified a length of less than 256 bytes.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

Trace types

Figure 2 on page 585, shows the overall structure of CS Linux. Each kernel-space trace type, relating to data transferred across a particular interface between CS Linux components, is shown in the diagram at the interface where it is traced.

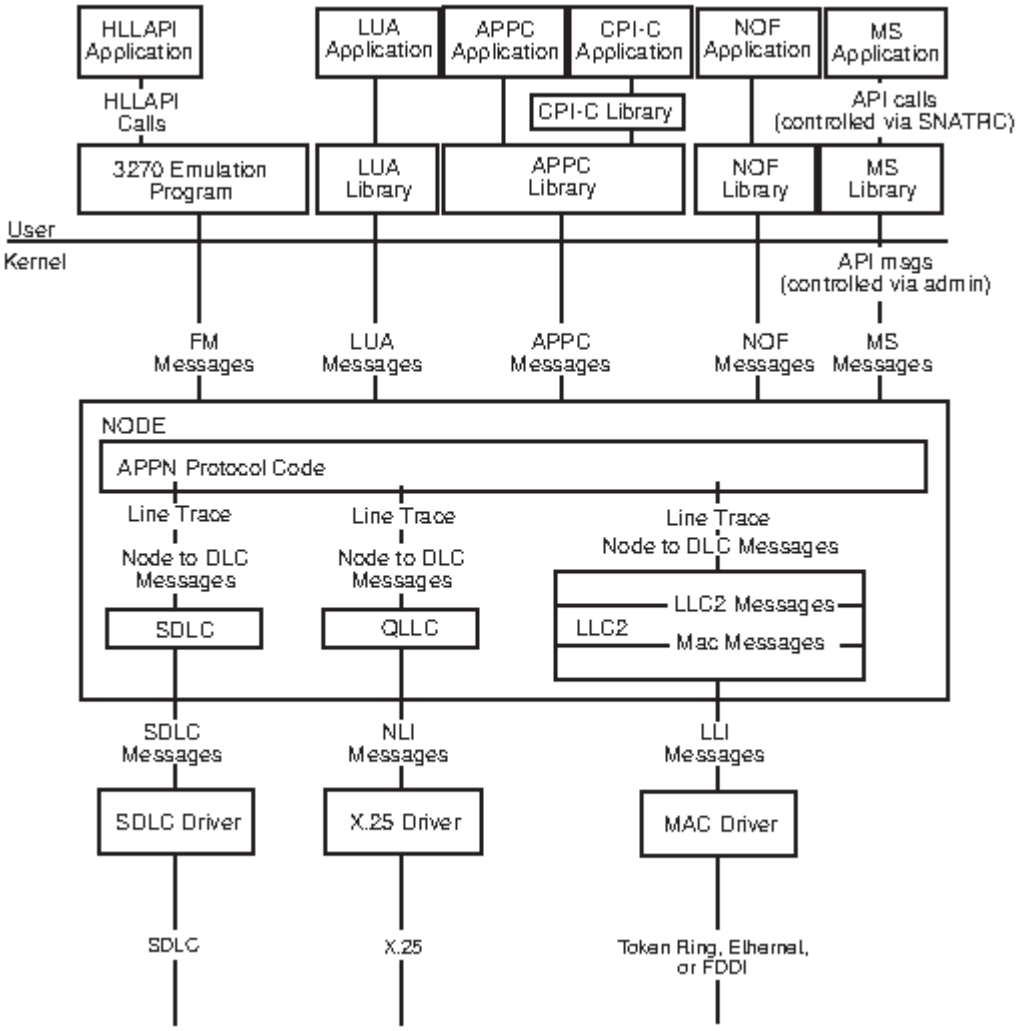

Figure 2. Overall Structure of CS Linux

Figure 2 on page 585 shows the following types of tracing, each of which can be controlled separately:

APPC messages
Messages between the APPC library and the node

LUA messages
Messages between the LUA library and the node

NOF messages
Messages between the NOF library and the node

MS messages
Messages between the MS library and the node

DLC line trace
SNA data sent on a DLC (tracing on these messages is controlled by the ADD_DLC_TRACE verb, not by SET_TRACE_TYPE)

Node to DLC messages
Messages between the APPN node and the DLC component

In addition, the following message types (internal to CS Linux) can be traced:

Node messages
Messages between components within the APPN protocol code

Control messages
Internal control messages between system components

スター・データ・ドル

START_DLC は、 DLC の活動化を要求します。

この verb は実行中のノードに対して発行する必要があります

VCB structure

```
typedef struct start_dlc
{
   AP_UINT16        opcode;                    /* verb operation code      */
   unsigned char    reserv2;                   /* reserved                 */
   unsigned char    format;                    /* reserved                 */
   AP_UINT16        primary_rc;                /* primary return code      */
   AP_UINT32        secondary_rc;              /* secondary return code    */
   unsigned char    dlc_name[8];               /* name of DLC              */
} START_DLC;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
追加されている DLC

dlc_name
開始する DLC の名前。 これは 8 バイトの ASCII ストリングで、名前が 8 バイトより短い場合に右側にスペースが埋め込まれます。これは、定義済みの DLC の名前と一致する必要があります。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameter:

primary_rc
AP_OK

This return code indicates only that the verb was issued successfully; the verb does not wait for the DLC to initialize, and so does not return error return codes if the initialization of the DLC fails. DLC initialization failures are reported using messages written to the error log file.

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc

ファイルの追加の削除
dlc_name パラメーターが定義済みの DLC の名前ではありませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: state check

If the verb does not execute because of a state error, CS Linux returns the following parameters.

primary_rc
AP_STATE_CHECK

secondary_rc

AP_DLC_DEACTIVATING
The specified DLC has already been started, and is in the process of being deactivated.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_STATE_CHECK, which are common to all NOF verbs.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

START_INTERNAL_PU

START_INTERNAL_PU requests DLUR to initiate SSCP-PU session activation for a previously defined local PU which is served by DLUR.

This verb must be issued to a running node.

VCB 構造体

```
typedef struct start_internal_pu
{
   AP_UINT16        opcode;                    /* verb operation code      */
   unsigned char    reserv2;                   /* reserved                 */
   unsigned char    format;                    /* reserved                 */
   AP_UINT16        primary_rc;                /* primary return code      */
   AP_UINT32        secondary_rc;              /* secondary return code    */
   unsigned char    pu_name[8];                /* internal PU name         */
   unsigned char    dlus_name[17];             /* DLUS name                */
   unsigned char    bkup_dlus_name[17];        /* Backup DLUS name         */
} START_INTERNAL_PU;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
AP_START_INTERNAL_PU

プール名
開始される内部 PU の名前 (これは、 DEFINE_INTERNAL_PU を使用して事前に定義されている必要があります) 。 この名前は、 8 バイトのタイプ A の EBCDIC ストリング (文字で始まる) で、右側に EBCDIC のスペースが埋め込まれます。

dlus_name
指定された PU に対する SSCP-PU セッションの活動化の活動化に DLUR が接続する DLUS ノードの名前。 この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

DEFINE_INTERNAL_PU verb で指定された DLUS を使用するか、 DEFINE_INTERNAL_PU で指定されていない場合は、 DEFINE_DLUR_DEFAULTS で指定されたグローバル・デフォルトを使用するには、このパラメーターを 17 の 2 進ゼロに設定してください。

bkup_dlus_name
指定された PU のバックアップ DLUS として DLUR が保管する DLUS ノードの名前。 この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

DEFINE_INTERNAL_PU verb で指定されたバックアップ DLUS 、または DEFINE_DLUR_DEFAULTS で指定されたグローバル・バックアップ・デフォルトを DEFINE_INTERNAL_PU で指定する場合は、このパラメーターを 17 の 2 進ゼロに設定してください。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc
可能な値は次のとおりです

ファイル名の入力が無効です
dlus_name パラメーターに、無効な文字が含まれているか、または正しい形式ではありませんでした。

アプインバリデータ・プクル名
bkup_dlus_name パラメーターに、無効な文字が含まれているか、または正しい形式ではありませんでした。

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc
可能な値は次のとおりです

デフォルトで定義されているアプリケーションがあります
この verb または DEFINE_INTERNAL_PU verb で DLUS 名が指定されておらず、DEFINE_DLUR_DEFAULTS verb が発行されていないため、デフォルトの DLUS が定義されていません。

付加が定義されていない
指定された PU 名は、 DEFINE_INTERNAL_PU を使用して定義された内部 PU の名前ではありませんでした。

AP_PU_ALREADY_アクティブ化中
PU は、すでに開始処理中です。

アクティブの追加 ALREADY_ACTIVE
PU はすでに開始されています。

戻りパラメーター: 失敗

verb が正常に実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
失敗

secondary_rc
可能な値は次のとおりです

拒否されたファイルの追加
DLUS がセッション活動化要求をリジェクトしました。

AP_DLUS_CAPS_ミスマッチ
構成された DLUS 名が DLUS ノードではありませんでした。

失敗した追加の ACTPU
ローカル・ノードは、 DLUS からのメッセージをリジェクトしました。 これは、内部エラー、リソース不足、または受信したメッセージの問題が原因である可能性があります。 CS Linux ログ・ファイルで、詳細情報を提供しているメッセージを確認してください。

Returned parameters: function not supported

If the verb does not execute because the node's configuration does not support it, CS Linux returns the following parameter:

primary_rc

AP_FUNCTION_NOT_SUPPORTED
The node does not support DLUR; this is defined by the *dlur_supported* parameter on DEFINE_NODE.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

開始日

START_LS は通常、非アクティブ LS を開始します あるいは、 LS を非アクティブにするために使用することもできますが、必要に応じて CS Linux によって自動的に活動化することも、リモート・システムによってアクティブにすることも指定できます。

注 : LS が専用 SDLC リンク、または QLLC PVC リンクである場合は、 CS Linux だけでなく、リモート・システムによって活動化される必要があります。 ノードの開始時に活動化する LS を定義し、障害後に自動的に再活動化するようにすることをお勧めします。これにより、リンクが常に使用可能であることを確認します。詳細については、 104 ページの『DEFINE_LS』 を参照してください。

この verb は実行中のノードに対して発行する必要があります

VCB structure

```
typedef struct start_ls
{
   AP_UINT16        opcode;           /* verb operation code                */
```

```
    unsigned char    reserv2;          /* reserved                            */
    unsigned char    format;           /* reserved                            */
    AP_UINT16        primary_rc;       /* primary return code                 */
    AP_UINT32        secondary_rc;     /* secondary return code               */
    unsigned char    ls_name[8];       /* name of link station                */
    unsigned char    enable;           /* start ls or enable auto-activation? */
    unsigned char    react_kicked;     /* retry in progress?                  */
    unsigned char    reserv3[2];       /* reserved                            */
} START_LS;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_START_LS

ls_name
Name of the link station to be started. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes, which must already have been defined by a DEFINE_LS verb.

enable
Specifies the action to be taken for the LS.

To start the LS, set this parameter to AP_ACTIVATE.

To leave the LS inactive but specify that it can be activated (either by CS Linux or by the remote system) when required, specify one or both of the following values (combined using a logical OR):

AP_AUTO_ACT
The LS can be activated automatically by CS Linux when required for a session. This value should be used only when the LS is defined to be auto-activatable (*auto_act_supp* in the LS definition is set to AP_YES); it re-enables auto-activation after the LS has been manually stopped using STOP_LS.

AP_REMOTE_ACT
The LS can 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 stopped, it will return to the defined setting.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・ チェックの追加

secondary_rc
可能な値は次のとおりです

追加されたファイルの名前が指定されています
ls_name パラメーターが、定義された LS の名前ではありませんでした。

リンクを使用可能にする機能
有効化 パラメーターが有効な値に設定されていませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・ チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: state check

If the verb does not execute because of a state error, CS Linux returns the following parameters.

primary_rc
AP_STATE_CHECK

secondary_rc
Possible values are:

AP_ACTIVATION_LIMITS_REACHED
The activation limits have been reached.

AP_LINK_DEACT_IN_PROGRESS
The specified LS is currently being deactivated. You cannot start it until the deactivation process has finished.

AP_ALREADY_STARTING
The specified LS is already starting.

AP_PARALLEL_TGS_NOT_SUPPORTED
A link to the remote system is already active. The adjacent node does not support parallel transmission groups.

AP_PORT_INACTIVE
The LS cannot be started because its associated port is not active.

react_kicked
Specifies whether CS Linux will retry the attempt to activate the LS (based on the *react_timer_retry* parameter in the LS definition). Possible values are:

AP_YES
LS activation will be retried (up to the number of attempts specified by *react_timer_retry*).

AP_NO
LS activation will not be retried.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_STATE_CHECK, which are common to all NOF verbs.

戻りパラメーター: 失敗

リモート・コンピューター上の SNA サブシステムに 接続できないために verb が正常に実行されなかった場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
LS_LS_失敗

secondary_rc
可能な値は次のとおりです

追加されたパートナーが見つかりません
この LS に関連付けられているポートから応答が受信されませんでした。 トークンリング、イーサネットの場合は、 LS 定義内の *mac_* アドレス パラメーターが正しいことを確認してください。

アプエラー
リモート・コンピューターへの接続を確立できませんでした。 これは、リモート・コンピューター上の SNA サブシステムが始動していないことが原因である可能性が LAN タイプ (トークンリング、イーサネット) 以外のリンク・タイプの場合は、 CS Linux が提供されたアドレス指定情報と一致するリモート・コンピューターを検出できなかったことを示している場合もあります。

Returned parameters: cancelled

If the verb does not execute because it was cancelled by another verb, CS Linux returns the following parameters:

primary_rc
AP_CANCELLED

secondary_rc
Possible values are:

AP_NO_SECONDARY_RC
A STOP_LS verb was issued before the START_LS verb had completed. The START_LS verb was cancelled.

AP_LINK_DEACTIVATED
The DLC or port used by the LS was stopped before the START_LS verb had completed. The START_LS verb was cancelled.

react_kicked
Specifies whether CS Linux will retry the attempt to activate the LS (based on the *react_timer_retry* parameter in the LS definition). Possible values are:

AP_YES
LS activation will be retried (up to the number of attempts specified by *react_timer_retry*).

AP_NO
LS activation will not be retried.

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

START_PORT

START_PORT requests the activation of a port. The DLC specified for the port must be active before this verb is issued.

This verb must be issued to a running node.

VCB 構造体

```
typedef struct start_port
{
   AP_UINT16        opcode;                    /* verb operation code      */
   unsigned char    reserv2;                   /* reserved                 */
   unsigned char    format;                    /* reserved                 */
   AP_UINT16        primary_rc;                /* primary return code      */
   AP_UINT32        secondary_rc;              /* secondary return code    */
   unsigned char    port_name[8];              /* name of port             */
} START_PORT;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_START_PORT

port_name
Name of port to be started. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes, which must already have been defined by a DEFINE_PORT verb.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: パラメーター・チェック

パラメーター・エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
パラメーター・チェックの追加

secondary_rc

ファイルの無効ポート
ポート名パラメーターが定義済みポートの名前ではありませんでした。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **パラメーター・チェックの追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: state check

If the verb does not execute because of a state error, CS Linux returns the following parameters.

primary_rc
AP_STATE_CHECK

secondary_rc
Possible values are:

AP_DLC_INACTIVE
The port cannot be started because its associated DLC is not active.

AP_DUPLICATE_PORT
The specified port has already been started.

AP_STOP_PORT_PENDING
The specified port is currently being deactivated. You cannot start it until the deactivation process has finished.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_STATE_CHECK, which are common to all NOF verbs.

戻りパラメーター: キャンセル済み

取り消されたために verb が実行されなかった場合、 CS Linux は以下のパラメーターを戻します。

primary_rc

キャンセル済み
この verb が完了する前に、 STOP_PORT verb が発行されました。 START_PORT verb が取り消されました。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

停止 DLC

STOP_DLC は、 DLC を停止するために CS Linux を要求します。これにより、 DLC を使用するアクティブ・ポートおよび LS も停止します。

この verb は実行中のノードに対して発行する必要があります

VCB 構造体

```
typedef struct stop_dlc
{
   AP_UINT16        opcode;                    /* verb operation code      */
   unsigned char    reserv2;                   /* reserved                 */
```

```
    unsigned char    format;                    /* reserved                 */
    AP_UINT16        primary_rc;                /* primary return code      */
    AP_UINT32        secondary_rc;              /* secondary return code    */
    unsigned char    stop_type;                 /* stop type                */
    unsigned char    dlc_name[8];               /* name of DLC              */
} STOP_DLC;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
AP_STOP_DLC

stop_type
必要な停止処理のタイプ。 可能な値は次のとおりです

ORDERDERLY_STOP
CS Linux は、 DLC の停止前にクリーンアップ操作を実行します。

即時 ATE_STOP
CS Linux は、 DLC を即時に停止します。

dlc_name
停止される DLC の名前。 これは 8 バイトの ASCII ストリングで、名前が 8 バイトより短い場合は右側にスペースが埋め込まれます。これは、 DEFINE_DLC verb によって既に定義されていなければなりません。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_DLC
The *dlc_name* parameter did not match the name of a defined DLC.

AP_UNRECOGNIZED_DEACT_TYPE
The *stop_type* parameter was not set to a valid value.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc

追加 STOP_DLC_PENDING
指定された DLC は、既に停止処理中です。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **状態検査の追加**に関連したさらに 2 次戻りコードがリストされます。

戻りパラメーター: キャンセル済み

取り消されたために verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc

キャンセル済み
stop_type パラメーターが規則正しい停止を指定しましたが、その DLC は、即時停止または障害状態を指定する 2 番目のコマンドによって停止されました。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

内部の停止 (_R)

STOP_INTERNAL_PU は、 DLUR がサービスを提供している以前に定義されたローカル PU に対して、SSCP-PU セッションの非活動化を開始するための DLUR を要求します。

この verb は実行中のノードに対して発行する必要があります

VCB 構造体

```
typedef struct stop_internal_pu
{
   AP_UINT16        opcode;                  /* verb operation code      */
   unsigned char    reserv2;                 /* reserved                 */
   unsigned char    format;                  /* reserved                 */
   AP_UINT16        primary_rc;              /* primary return code      */
   AP_UINT32        secondary_rc;            /* secondary return code    */
   unsigned char    pu_name[8];              /* internal PU name         */
   unsigned char    stop_type;               /* type of stop requested   */
} STOP_INTERNAL_PU;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
AP_STOP_INTERNAL_PU

プール名
SSCP-PU セッションが非活動化される内部 PU の名前。 これは、 8 バイトのタイプ A の EBCDIC ストリング (文字で始まる) で、右側に EBCDIC のスペースが埋め込まれます。

stop_type
PU の停止方法を指定します。 可能な値は次のとおりです

ORDERDERLY_STOP
SSCP-PU セッションを非活動化する前に、基礎となる PLU-SLU セッションおよび SSCP-LU セッションをすべて非活動化してください。

即時 ATE_STOP
SSCP-PU セッションを即時に非活動化します。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc

AP_INVALID_STOP_TYPE
The *stop_type* parameter was not set to a valid value.

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc
可能な値は次のとおりです

付加が定義されていない
指定された PU 名が、定義された内部 PU の名前と一致しませんでした。

AP_PU_ALREADY_DE アクティブにしている
PU は既に停止処理中です。

付加 PU_NOT_ACTIVE
PU はアクティブではありません。

戻りパラメーター: 関数はサポートされません

ノードの構成がそれをサポートしていないために verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc

付加機能がサポートされていません
ノードは DLUR をサポートしていません。これは、 DEFINE_NODE の *dlur_supported* パラメーターによって定義されます。

Returned parameters: other conditions

Appendix B, “共通戻りコード,” on page 665 lists further combinations of primary and secondary return codes that are common to all NOF verbs.

停止 (LS)

STOP_LS は、アクティブな LS を停止します。 あるいは、非アクティブ LS に対して発行することもできます。これは、リモート・システムによって必要または活動化されたときに、 LS が CS Linux によって自動的に活動化できないことを指定するためです。これらの両方が使用不可の場合、 LS は START_LS を発行することによってのみアクティブにできます。

この verb は実行中のノードに対して発行する必要があります

VCB 構造体

```
typedef struct stop_ls
{
   AP_UINT16        opcode;           /* verb operation code              */
   unsigned char    reserv2;          /* reserved                         */
   unsigned char    format;           /* reserved                         */
   AP_UINT16        primary_rc;       /* primary return code              */
   AP_UINT32        secondary_rc;     /* secondary return code            */
   unsigned char    stop_type;        /* stop type                        */
```

```
    unsigned char    ls_name[8];        /* name of link station               */
    unsigned char    disable;           /* disable remote or auto activation? */
    unsigned char    reserved[3];       /* reserved                          */
} STOP_LS;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
AP_STOP_LS

stop_type
必要な停止処理のタイプ。 可能な値は次のとおりです

ORDERDERLY_STOP
CS Linux は LS を停止する前にクリーンアップ操作を実行します。

即時 ATE_STOP
CS Linux は、直ちに LS を停止します。

ls_name
停止する LS の名前。 これは 8 バイトの ASCII ストリングで、名前が 8 バイトより短い場合は右側にスペースが埋め込まれます。これは、 DEFINE_LS verb によって既に定義されている必要があります。

無効化
LS に対して実行されるアクションを指定します。

アクティブ LS を停止し、自動活動化およびリモート活動化のデフォルト設定に戻るには、このパラメーターを **アプ・ ノー**に設定します。

非アクティブ LS を CS Linux によって活動化できないか、またはリモート・システムによって活動化できないことを指定するには、以下の値のいずれかまたは両方を指定します (論理 **それとも**を使用して結合) 。

自動実行
LS は CS Linux によって自動的に活動化できません。

追加リモート・ アクション
LS はリモート・システムによって活動化できません。 この値は、 LS 定義内の *unable_remote_act* の定義済み値を変更しません。 LS が次に開始および停止されたときに、定義された設定値に戻ります。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_LINK_NOT_DEFD
The *ls_name* parameter did not match the name of a defined LS.

AP_UNRECOGNIZED_DEACT_TYPE
The *stop_type* parameter was not set to a valid value.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc

リンクの非アクティブ化が進行中
指定された LS は、既に非活動化の処理中です。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **状態検査の追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: cancelled

If the verb does not execute because it was cancelled, CS Linux returns the following parameters.

primary_rc

AP_CANCELLED
The *stop_type* parameter specified an orderly stop, but the LS was then stopped by a second verb specifying an immediate stop or by a failure condition.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

停止ポート

STOP_PORT を使用すると、アプリケーションはポートを停止できます。 これにより、ポートを使用しているアクティブな LSs もすべて停止します。

この verb は実行中のノードに対して発行する必要があります

VCB structure

```
typedef struct stop_port
{
   AP_UINT16        opcode;                  /* verb operation code      */
   unsigned char    reserv2;                 /* reserved                 */
   unsigned char    format;                  /* reserved                 */
   AP_UINT16        primary_rc;              /* primary return code      */
   AP_UINT32        secondary_rc;            /* secondary return code    */
   unsigned char    stop_type;               /* Stop Type                */
   unsigned char    port_name[8];            /* name of port             */
} STOP_PORT;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
AP_STOP_PORT

stop_type
必要な停止処理のタイプ。 可能な値は次のとおりです

ORDERDERLY_STOP
CS Linux は、ポートを停止する前にクリーンアップ操作を実行します

即時 ATE_STOP
CS Linux は、ポートを即時に停止します。

ポート名
停止するポートの名前。 これは 8 バイトの ASCII ストリングで、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。

戻りパラメーター: 正常に実行されたパラメーター

verb が正常に実行されると、 CS Linux は以下のパラメーターを戻します。

primary_rc
アプオク

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Possible values are:

AP_INVALID_PORT_NAME
The *port_name* parameter did not match the name of a defined port.

AP_UNRECOGNIZED_DEACT_TYPE
The *stop_type* parameter was not set to a valid value.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: 状態チェック

状態エラーが原因で verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc

AP_STOP_PORT_PENDING
指定されたポートは、既に非活動化の処理中です。

665 ページの『付録 B 共通戻りコード』 には、すべての NOF verb に共通な、 **状態検査の追加**に関連したさらに 2 次戻りコードがリストされます。

Returned parameters: cancelled

If the verb does not execute because it has been cancelled, CS Linux returns the following parameters:

primary_rc

AP_CANCELLED
The *stop_type* parameter specified an orderly stop, but the port was then stopped by a second verb specifying an immediate stop or by a failure condition.

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

TERM_NODE

TERM_NODE allows the application to stop the node with a specified urgency. This also stops all connectivity resources associated with the node.

This verb must be issued to a running node.

VCB structure

```
typedef struct term_node
{
   AP_UINT16        opcode;                 /* verb operation code      */
   unsigned char    reserv2;                /* reserved                 */
   unsigned char    format;                 /* reserved                 */
   AP_UINT16        primary_rc;             /* primary return code      */
   AP_UINT32        secondary_rc;           /* secondary return code    */
   unsigned char    stop_type;              /* stop type                */
} TERM_NODE;
```

提供されるパラメーター

アプリケーションは以下のパラメーターを提供します。

オペコード
ノードへの追加 （ ノード ）

stop_type
CS Linux がノードを停止する方法を指定します。 可能な値は次のとおりです

アプアアボート
クリーンアップ処理を行わずに即時に停止します。 この値は、重大なエラー条件でのみ使用する必要があります。これは、ノードのリソースを使用する他のプログラムに問題が生じる可能性があるため

追加シャットダウン
停止する前に、ノードに関連したすべての LPAR を非活動化します。

静止の解除
ノードが静止しているネットワークに指示し、すべてのモードのセッション限度をリセットし、ノードの LU のすべてのエンドポイント・セッションをアンバインドしてから、 **追加シャットダウン**の場合と同様に停止します。

静止 - ISR
静止の解除と同じ機能。ただし、すべての中間セッションが終了するのをノードが待機する点が異なります。 この値はネットワーク・ノードにのみ適用されます

エイプの退化 （ クリーン ）
静止の解除と同じ機能。ただし、セッション限度はリセットされず、 RTP 接続は、 LSs が非活動化される前に正常に終了することが許可されます。

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

UNREGISTER_INDICATION_SINK

UNREGISTER_INDICATION_SINK unregisters the NOF application so that it no longer receives indications of a particular type (previously specified using REGISTER_INDICATION_SINK).

If the application has registered more than once to accept multiple indication types, it must unregister separately for each indication that it no longer wants to receive.

This verb must always be issued using the asynchronous NOF API entry point, including the callback routine that was supplied on the REGISTER_INDICATION_SINK verb (for more information about the asynchronous NOF API entry point, see “Asynchronous entry point: nof_async” on page 21).

This verb may be issued to the domain configuration file, to a running node or to a server where the node is not running, or to the `sna.net` file, depending on the type of indication for which the application is unregistering.

VCB 構造体

```
typedef struct unregister_indication_sink
{
   AP_UINT16        opcode;             /* verb operation code              */
   unsigned char    reserv2;            /* reserved                         */
   unsigned char    format;             /* reserved                         */
   AP_UINT16        primary_rc;         /* primary return code              */
   AP_UINT32        secondary_rc;       /* secondary return code            */
   AP_UINT32        proc_id;            /* reserved                         */
   AP_UINT16        queue_id;           /* reserved                         */
   AP_UINT16        indication_opcode;  /* opcode of indication to be unsunk */
} UNREGISTER_INDICATION_SINK;
```

Supplied parameters

The application supplies the following parameters:

opcode
AP_UNREGISTER_INDICATION_SINK

indication_opcode
The *opcode* parameter of the indication that is no longer required.

Returned parameters: successful execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

Returned parameters: parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc

AP_INVALID_OP_CODE
The *indication_opcode* parameter did not match the *opcode* of any of the CS Linux NOF indications, or the application has not previously registered to receive the specified indication on this target handle.

Appendix B, “共通戻りコード,” on page 665 lists further secondary return codes associated with AP_PARAMETER_CHECK, which are common to all NOF verbs.

戻りパラメーター: 関数はサポートされません

ローカル・ノードが指定された指示に関連する機能をサポートしていないために verb が正常に実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc

付加機能がサポートされていません
ローカル・ノードは指定された指示をサポートしていません。 それぞれの指示に必要なサポートの詳細については、 603 ページの『第 4 章 NOF Indications』の各指示の説明を参照してください。

戻りパラメーター: その他の条件

665 ページの『付録 B 共通戻りコード』 は、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードの組み合わせをさらにリストします。

Chapter 4. NOF Indications

This chapter provides the following information for each NOF indication:

- Description of the indication's purpose and usage
- Verb control block (VCB) structure, as defined in the NOF API header file `/usr/include/sna/nof_c.h` (AIX) or `/opt/ibm/sna/include/nof_c.h` (Linux)
- Explanations of the parameters returned to the application in the VCB

For information about how the application registers to receive NOF indications, see “登録受信側のシンク” on page 557.

構成の指示

この指示が生成されるのは、別の NOF アプリケーションまたは CS Linux 管理ツールがターゲット構成を変更した場合、ターゲット・ノードが停止または開始されたとき、またはターゲット・ノードが所有する DLC、ポート、または LS が停止または開始されたときです。 ターゲットは、 CS Linux ソフトウェアが実行されているサーバー上のドメイン構成ファイル、実行中ノード、または非アクティブ・ノードにすることができます。 このターゲットは、この指示を受け取るために登録する REGISTER_INDICATION_SINK verb の *target_handle* パラメーターによって識別されます。

VCB structure

No specific VCB structure is associated with this indication. To register for configuration indications, the application specifies the value `AP_CONFIG_INDICATION` as the *indication_opcode* parameter on REGISTER_INDICATION_SINK. When a change is made, CS Linux then reports this to the application's callback routine by sending a copy of the VCB from the NOF verb that made the change. For example, if the configuration was modified by a DEFINE_DLC verb, CS Linux sends a copy of the DEFINE_DLC VCB to the application's callback routine.

To enable the application to distinguish between configuration indications and asynchronous responses to its own NOF verbs, CS Linux changes the *primary_rc* parameter in the VCB for a configuration indication. The value `AP_INDICATION` identifies a VCB associated with a configuration indication; the value `AP_OK`, or any other value, indicates an asynchronous response to one of the application's own NOF verbs.

The following events are not reported as configuration indications:

- Changes to the SNA network file `sna.net`. To receive indications of these changes, the application must register for the indication type `AP_SNA_NET_INDICATION`. For more information, see “SNA_NET_INDICATION” on page 653.
- Starting and stopping the SNA software on other servers. To receive indications of these changes, the application must register for the indication type `AP_SERVER_INDICATION`. For more information, see “SERVER_INDICATION” on page 648.

The range of VCBs that can be returned as configuration indications depends on the type of target handle specified on REGISTER_INDICATION_SINK:

Domain configuration file
The application can receive VCBs for any verbs that modify domain resources but not node resources (verbs that can be issued to the domain configuration file).

Node configuration file
The application can receive VCBs for any verbs that modify node resources.

Running node
The application can receive VCBs for any verbs that modify node resources, TERM_NODE VCBs, and START_* and STOP_* VCBs for DLCs, ports, and LSs.

Inactive node
The application can receive VCBs for any verbs that modify node resources and also INIT_NODE VCBs.

DIRECTORY_兆し

この指示は、ローカル・ディレクトリー・データベースにエントリーが追加されたり、ローカル・ディレクトリー・データベースに追加されたりした場合

VCB structure

```
typedef struct directory_indication
{
   AP_UINT16        opcode;              /* verb operation code              */
   unsigned char    reserv2;             /* reserved                         */
   unsigned char    format;              /* reserved                         */
   AP_UINT16        primary_rc;          /* primary return code              */
   AP_UINT32        secondary_rc;        /* secondary return code            */
   unsigned char    data_lost;           /* previous indication lost         */
   unsigned char    removed;             /* is entry being removed?          */
   unsigned char    resource_name[17];   /* resource name                    */
   unsigned char    invalid;             /* invalid entry being removed?     */
   AP_UINT16        resource_type;       /* resource type                    */
   unsigned char    parent_name[17];     /* parent resource name             */
   unsigned char    entry_type;          /* type of the directory entry      */
   AP_UINT16        parent_type;         /* parent resource type             */
   unsigned char    description[32];     /* resource description             */
   unsigned char    reserv3[16];         /* reserved                         */
   AP_UINT16        real_owning_cp_type;     /* CP type of real owner        */
   unsigned char    real_owning_cp_name[17]; /* CP name of real owner        */
   AP_UINT16        supplier_cp_type;        /* CP type of supplier          */
   unsigned char    supplier_cp_name[17];    /* CP name of supplier          */
   unsigned char    reserva;                 /* reserved                     */
} DIRECTORY_INDICATION;
```

パラメーター

オペコード
AP_DIRECTORY_指標

primary_rc
アプオク

データ損失
前のディレクトリー指示が失われたかどうかを指定します。 CS Linux は、標識を送信できない状態 (例えば、内部リソース不足) を検出すると、条件がクリアされた後に次の指示に データ損失 パラメーターを設定することによって、これを示します。 可能な値は次のとおりです

類人猿
以前の 1 つ以上のディレクトリー指示が失われました。 この VCB の後のフィールドは、ゼロに設定することができます。

アプ・ノー
以前のディレクトリー指示は失われませんでした。

削除
指定されたリソースがディレクトリーから除去されたか、またはディレクトリーに追加されたかを指定します。 可能な値は次のとおりです

類人猿
エントリーは削除されました。

アプ・ノー
エントリーが追加されました。

リソース名
リソースの完全修飾名。 この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

無効
エンド・ノードがそのリソースをネットワーク・ノードに登録すると、これらのリソースのために新しいディレクトリー項目がネットワーク・ノードのディレクトリー・データベースに追加されます。これらのいずれかのリソースについてデータベースに既に明示的に定義された項目が含まれているが、その項目が登録済みリソースと一致しない場合、 CS Linux は無効な項目を除去し、それを正しい項目で置き換えます。 このパラメーターは , 項目が正しくないために除去されたか , 登録された資源から正しい項目に置き換えられたか , あるいは明示的に削除されたために , 除去されたかどうかを示すために使用されます。 可能な値は次のとおりです

類人猿
エントリーが正しくないため、削除されました。

アプ・ノー
この項目は、明示的に削除されたため、除去されました。

ローカル・ノードがネットワーク・ノードでない場合、または 削除 が **アプ・ノー**に設定されている場合、このパラメーターは使用されません。

リソース・タイプ
リソース・タイプ。 可能な値は次のとおりです

リソースの追加
ネットワーク・ノード。

AP_ENCP_RESOURCE
ノードの終了。

AP_LU_RESOURCE
ルゥ

parent_name
親リソースの完全修飾名。 リソース・タイプ が **リソースの追加**の場合、これは 17 の 2 進ゼロに設定されます。

この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

エントリー・タイプ
ディレクトリー項目のタイプを指定します。 これは、以下のいずれかです。

アプアホーム
ローカル・リソース。

AP_CACHE
キャッシュされた項目。

登録の登録
登録済みリソース (NN のみ)

parent_type
登録されるリソースの親タイプを指定します。 リソース・タイプ が **リソースの追加**の場合、このパラメーターは使用されません。 可能な値は次のとおりです

リソースの追加
ネットワーク・ノード。

AP_ENCP_RESOURCE
ノードの終了。

記述
リソースを記述するヌル終了テキスト・ストリング (リソースの定義で指定されたもの) 。

所有者 *cp_type* の再レルム
このパラメーターは、ローカル・ノードがネットワーク・ノードまたは分岐ネットワーク・ノードである場合にのみ適用されます。それ以外の場合は予約済みです。

このディレクトリー項目によって識別されるリソースを所有する実 CP が、親リソースであるか、別のノードであるかを指定します。 これは、以下のいずれかです。

追加なし
実際の所有者は、親リソースです。

AP_ENCP_RESOURCE
実際の所有者は、親リソースではないエンド・ノードです。 例えば、リソースが分岐ネットワーク・ノード (BrNN) のドメイン内のエンド・ノードによって所有されている場合、この BrNN のネットワーク・ノード・サーバーのディレクトリーには、親リソースとしての BrNN が含まれますが、実際の所有 CP はエンド・ノードです。

再割り当て *cp_name*
このパラメーターは、ローカル・ノードがネットワーク・ノードまたは分岐ネットワーク・ノードである場合にのみ適用されます。それ以外の場合は予約済みです。

所有者 *cp_type* の再レルム パラメーターが、リソースの実所有者が親ではないことを示している場合、このパラメーターは、リソースを所有する CP の完全修飾名を指定します。それ以外の場合は、予約されます。

この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、 1 から 8 文字の A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および 1 から 8 文字の A ストリング文字のネットワーク名で構成されます。

サプライ *cp_type*
このパラメーターは、ローカル・ノードがネットワーク・ノードまたは分岐ネットワーク・ノードである場合にのみ適用されます。それ以外の場合は予約済みです。

このディレクトリー項目が、リソースの所有 CP ではない別のノードによって登録されたかどうかを指定します。 これは、以下のいずれかです。

追加なし
ディレクトリー項目が登録されていないか、または所有する CP によって登録されています。

AP_ENCP_RESOURCE
ディレクトリー項目は、所有している CP ではないノードによって登録されました。 例えば、リソースが、ローカル・ノードのドメイン内にある分岐ネットワーク・ノード (BrNN) のドメイン内のエンド・ノードによって所有されている場合、 BrNN はそのリソースをローカル・ノードに登録するため、サプライヤーです。ただし、実際の所有 CP はエンド・ノードです。

補助 *cp_name*
このパラメーターは、ローカル・ノードがネットワーク・ノードまたは分岐ネットワーク・ノードである場合にのみ適用されます。それ以外の場合は予約済みです。

サプライ *cp_type* パラメーターが、ディレクトリー項目が所有リソースではないノードによって登録されたことを示している場合、このパラメーターは、登録を提供した CP の完全修飾名を指定します。それ以外の場合は、予約されます。

この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、 1 から 8 文字の A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および 1 から 8 文字の A ストリング文字のネットワーク名で構成されます。

DLC_指標

この指示は、 DLC の状態が活動状態と非アクティブの間で状態を変更すると

VCB structure

```
typedef struct dlc_indication
{
   AP_UINT16        opcode;             /* verb operation code             */
   unsigned char    reserv2;            /* reserved                        */
   unsigned char    format;             /* reserved                        */
   AP_UINT16        primary_rc;         /* primary return code             */
   AP_UINT32        secondary_rc;       /* secondary return code           */
   unsigned char    data_lost;          /* previous indication lost        */
   unsigned char    deactivated;        /* has session been deactivated?   */
   unsigned char    dlc_name[8];        /* link station name               */
   unsigned char    description[32];    /* resource description            */
   unsigned char    reserv1[16];        /* reserved                        */
```

```
   unsigned char    reserva[20];        /* reserved                         */
} DLC_INDICATION;
```

パラメーター

オペコード
追加 DLC_指標

primary_rc
アプオク

データ損失
前の指示が失われたかどうかを指定します。 CS Linux は、標識を送信できない状態 (例えば、内部リソース不足) を検出すると、条件がクリアされた後に次の指示に データ損失 パラメーターを設定することによって、これを示します。 可能な値は次のとおりです

類人猿
1 つ以上の直前の DLC 指示が失われました。 この VCB の後のフィールドは、ゼロに設定することができます。

アプ・ノー
以前の DLC 指示は失われませんでした。

非活動化
DLC が非アクティブになったか、アクティブになったかを指定します。 可能な値は次のとおりです

類人猿
DLC は非活動状態になりました。

アプ・ノー
DLC がアクティブになりました。

dlc_name
DLC の名前。 これは 8 バイトの ASCII ストリングで、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。

記述
DLC の定義に指定されている、 DLC を記述するヌルで終了するテキスト・ストリング。

DLUR_LU_指標

この指示は、 DLUR LU が活動化または非活動化されると生成されます この指示は、登録済みのアプリケーションが、現在アクティブな DLUR LU のリストを保守するために使用できます。

VCB structure

```
typedef struct dlur_lu_indication
{
   AP_UINT16        opcode;             /* Indication operation code        */
   unsigned char    reserv2;            /* reserved                         */
   unsigned char    format;             /* reserved                         */
   AP_UINT16        primary_rc;         /* primary return code              */
   AP_UINT32        secondary_rc;       /* secondary return code            */
   unsigned char    data_lost;          /* Previous indication lost?        */
   unsigned char    reason;             /* reason for this indication       */
   unsigned char    lu_name[8];         /* LU name                          */
   unsigned char    pu_name[8];         /* PU name                          */
   unsigned char    nau_address;        /* NAU address                      */
   unsigned char    reserv5[7];         /* reserved                         */
} DLUR_LU_INDICATION;
```

Parameters

opcode
AP_DLUR_LU_INDICATION

primary_rc
AP_OK

data_lost
Specifies whether any previous directory indications have been lost. If CS Linux detects a condition that prevents it from sending an indication (for example an internal resource shortage), it indicates this by setting the *data_lost* parameter on the next indication after the condition has cleared. Possible values are:

AP_YES
One or more previous directory indications were lost. Later fields in this VCB may be set to zeros.

AP_NO
No previous directory indications were lost.

reason
Reason for this indication. Possible values are:

AP_ADDED
The DLUR has just been activated by the DLUS.

AP_REMOVED
The DLUR has been deactivated, either explicitly by the DLUS or implicitly by a link failure or the deactivation of the PU.

lu_name
Name of the logical unit (LU). This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces.

pu_name
Name of the physical unit (PU) that this LU uses. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces.

nau_address
Network accessible unit (NAU) address of the LU. This value must be in the range 1-255.

DLUR_PU_指標

この指示は、従属 LU リクエスター (DLUR) 機能をサポートするノードの物理装置 (PU) が活動化しようとしているとき、活動化、活動化、活動化、または非活動化に失敗した場合に生成されます。 この指示は、現在アクティブな DLUR PU のリストを維持するために使用することができます。

VCB structure

```
typedef struct dlur_pu_indication
{
   AP_UINT16        opcode;              /* Indication operation code        */
   unsigned char    reserv2;             /* reserved                         */
   unsigned char    format;              /* reserved                         */
   AP_UINT16        primary_rc;          /* primary return code              */
   AP_UINT32        secondary_rc;        /* secondary return code            */
   unsigned char    data_lost;           /* Previous indication lost?        */
   unsigned char    reason;              /* reason for this indication       */
   unsigned char    pu_name[8];          /* PU name                          */
   unsigned char    pu_id[4];            /* PU identifier                    */
   unsigned char    pu_location;         /* downstream or local PU           */
   unsigned char    pu_status;           /* status of the PU                 */
   unsigned char    dlus_name[17];       /* current DLUS name                */
   unsigned char    dlus_session_status; /* status of the DLUS pipe          */
   unsigned char    reserv5[2];          /* reserved                         */
} DLUR_PU_INDICATION;
```

パラメーター

オペコード
ID の追加の指示

primary_rc
アプオク

データ損失
前のディレクトリー指示が失われたかどうかを指定します。 CS Linux は、標識を送信できない状態 (例えば、内部リソース不足) を検出すると、条件がクリアされた後に次の指示に データ損失 パラメーターを設定することによって、これを示します。 可能な値は次のとおりです

類人猿
以前の 1 つ以上のディレクトリー指示が失われました。 この VCB の後のフィールドは、ゼロに設定することができます。

アプ・ノー
以前のディレクトリー指示は失われませんでした。

理由
指示の原因。 可能な値は次のとおりです

可能な値は次のとおりです

追加アクティビティーが開始されました
PU は活動化中です。

非アクティブ化
PU が活動状態になりました。

非アクティブ化中
PU は非活動状態になりました。

失敗
PU は失敗しました。

追加の活動化が失敗しました
PU は活動化に失敗しました。

プール名
物理装置 (PU) の名前。 これは、 8 バイトの英数字のタイプ A の EBCDIC ストリング (文字で始まる) で、右側に EBCDIC のスペースが埋め込まれます。

パブリッシュ *ID*
DEFINE_INTERNAL_PU verb で定義されているか、またはダウンストリーム PU から XID で取得された PU ID 。 この値は、 4 バイトの 16 進数ストリングです。 ビット 0 から 11 は、ブロック番号に設定され、ビット 12 から 31 は、 PU を一意的に識別する ID 番号に設定されます。

プール・ロケーション
PU のロケーション。 可能な値は次のとおりです

内部
この PU は、 DEFINE_INTERNAL_PU verb によって定義されています。

ダウンストリーム
PU はダウンストリーム・コンピューターにあります。

pu_status
DLUR から見た、 PU の状況。 可能な値は次のとおりです

RESET_NO_RETRY
PU はリセット状態にあり、再試行されません。

再設定再試行
PU はリセット状態にあり、再試行されます。

アプペン D_アクプー
PU はホストからの ACTPU を待機しています。

付加された ACTPU_RSP
DLUR は、 ACTPU を PU に転送した後、 PU がその PU に応答するのを待っています。

アクティブ (アクティブ)
PU は活動状態です。

付加された DACTPU_RSP
DLUR は、 DACTPU を PU に転送した後、 PU がその PU に応答するのを待っています。

付加された 1 番目の操作
DLUR は、 PU を非活動化する前に、必要なすべてのイベントの完了を待っています。

dlus_name
PU が現在使用している従属 LU サーバー (DLUS) ノードの名前 (または使用しようとしている) 。 この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。 理由 パラメーターが **失敗**に設定されている場合、 *dlus_name* パラメーターはすべてゼロに設定されます。

dlus_session_status
PU によって現在使用されている DLUS パイプの状況。 可能な値は次のとおりです

アクティブの追加 （ _L ）
DLUS パイプは現在活動化中です。

アクティブ （ アクティブ ）
DLUS パイプがアクティブです。

年金を非アクティブにする
DLUS パイプは現在非活動化されています。

非アクティブ
DLUS パイプが非アクティブです。

DLUS_指標

この指示は、 DLUS ノードへのパイプがアクティブと 非アクティブの間で状態を変更すると生成されます。 パイプが非アクティブになると 、その指示にはパイプ統計も含まれます。

VCB structure

```
typedef struct dlus_indication
{
   AP_UINT16        opcode;                 /* verb operation code          */
   unsigned char    reserv2;                /* reserved                     */
   unsigned char    format;                 /* reserved                     */
   AP_UINT16        primary_rc;             /* primary return code          */
   AP_UINT32        secondary_rc;           /* secondary return code        */
   unsigned char    data_lost;              /* previous indication lost     */
   unsigned char    deactivated;            /* has DLUS become inactive?    */
   unsigned char    dlus_name[17];          /* DLUS name                    */
   unsigned char    reserv1;                /* reserved                     */
   PIPE_STATS       pipe_stats;             /* pipe statistics              */
   unsigned char    persistent_pipe_support; /* reserved                    */
   unsigned char    persistent_pipe;        /* reserved                     */
   unsigned char    reserva[18];            /* reserved                     */
} DLUS_INDICATION;
```

```
typedef struct pipe_stats
{
   AP_UINT32       reqactpu_sent;           /* REQACTPUs sent to DLUS       */
   AP_UINT32       reqactpu_rsp_received;   /* RSP(REQACTPU)s received      */
                                            /*  from DLUS                   */
   AP_UINT32       actpu_received;          /* ACTPUs received from DLUS    */
   AP_UINT32       actpu_rsp_sent;          /* RSP(ACTPU)s sent to DLUS     */
   AP_UINT32       reqactpu_sent;           /* REQDACTPUs sent to DLUS      */
   AP_UINT32       reqdactpu_rsp_received;  /* RSP(REQDACTPU)s received     */
                                            /*  from DLUS                   */
   AP_UINT32       dactpu_received;         /* DACTPUs received from DLUS   */
   AP_UINT32       dactpu_rsp_sent;         /* RSP(DACTPU)s sent to DLUS    */
   AP_UINT32       actlu_received;          /* ACTLUs received from DLUS    */
   AP_UINT32       actlu_rsp_sent;          /* RSP(ACTLU)s sent to DLUS     */
   AP_UINT32       dactlu_received;         /* DACTLUs received from DLUS   */
   AP_UINT32       dactlu_rsp_sent;         /* RSP(DACTLU)s sent to DLUS    */
   AP_UINT32       sscp_pu_mus_rcvd;        /* MUs for SSCP-PU sessions rcvd */
   AP_UINT32       sscp_pu_mus_sent;        /* MUs for SSCP-PU sessions sent */
   AP_UINT32       sscp_lu_mus_rcvd;        /* MUs for SSCP-LU sessions rcvd */
```

```
   AP_UINT32       sscp_lu_mus_sent;        /* MUs for SSCP-LU sessions sent */
} PIPE_STATS;
```

パラメーター

オペコード
追加の DLUS_**指標**

primary_rc
アプオク

データ損失
直前の DLUS 指示が失われたかどうかを指定します。 CS Linux は、標識を送信できない状態 (例えば、内部リソース不足) を検出すると、条件がクリアされた後に次の指示に データ損失 パラメーターを設定することによって、これを示します。 可能な値は次のとおりです

類人猿
1 つ以上の直前のダウンストリーム LU 指示が失われました。 この VCB の後のフィールドは、ゼロに設定することができます。

アプ・ノー
以前のダウンストリーム LU 指示は失われませんでした。

非活動化
パイプが非アクティブになったか、アクティブになったかを指定します。 可能な値は次のとおりです

類人猿
パイプが非アクティブになりました。

アプ・ノー
パイプがアクティブになりました。

dlus_name
DLUS ノードの名前。 この名前は 17 バイトの EBCDIC ストリングで、右側には EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

パイプが非活動化されている場合は、 **ピペットスタット** 構造が組み込まれます。 この構造体のフィールドは以下のとおりです。

pipe_stats.reqactpu_sent
パイプ上で DLUS に送信された REQACTPU の数。

pipe_stats.reqactpu_rsp_受信済み
パイプ上で DLUS から受信した RSP(REQACTPU)s の数。

pipe_stats.actpu_受け入れた
パイプ上で DLUS から受信した ACTPU の数。

***pipe_stats.actpu_rsp_sent* 送信済み**
パイプ上で DLUS に送信された RSP(ACTPU)s の数。

pipe_stats.reqdactpu_sent
パイプ上で DLUS に送信された REQDACTPU の数。

pipe_stats.reqdactpu_rsp_受信済み
パイプ上で DLUS から受信した RSP(REQDACTPU)s の数。

pipe_stats.dactpu_受け入れた
パイプ上の DLUS から受信した DACTPU の数。

***pipe_stats.dactpu_rsp_sent* 送信済み**
パイプ上で DLUS に送信された RSP(DACTPU)s の数。

***pipe_stats.actlu_* 受信済み**
パイプ上で DLUS から受信した ACTLU の数。

pipe_stats.actlu_rsp_sent
パイプ上で DLUS に送信された RSP(ACTLU)s の数。

pipe_stats.dactlu_受領済み
パイプ上で DLUS から受信した DACTLU の数。

pipe_stats.dactlu_rsp_sent
パイプ上で DLUS に送信された RSP(DACTLU)s の数。

pipe_stats.sscp_pu_mus_rcvd
パイプ上で DLUS から受信した SSCP-PU CU の数。

pipe_stats.sscp_pu_mus_sent
パイプ上で DLUS に送信された SSCP-PU MU の数。

pipe_stats.sscp_lu_mus_rcvd
パイプ上で DLUS から受信した SSCP-LU MU の数。

pipe_stats.sscp_lu_mus_sent
パイプ上で DLUS に送信された SSCP-LU MU の数。

ダウンロードされたルートの指標

この指示は、 LU-SSCP セッションまたはダウンストリーム LU 間の PLU-SLU セッションのどちらか一方で、アクティブと非アクティブの間で状態が変更された場合に生成されます。 これらのセッションのいずれかが非アクティブになると、そのセッションのセッション統計も表示されます。

VCB 構造体

```
typedef struct downstream_lu_indication
{
   AP_UINT16       opcode;               /* verb operation code             */
   unsigned char   reserv2;              /* reserved                        */
   unsigned char   format;               /* reserved                        */
   AP_UINT16       primary_rc;           /* primary return code             */
   AP_UINT32       secondary_rc;         /* secondary return code           */
   unsigned char   data_lost;            /* previous indication lost        */
   unsigned char   dspu_name[8];         /* PU name                         */
   unsigned char   ls_name[8];           /* Link station name               */
   unsigned char   dslu_name[8];         /* LU name                         */
   unsigned char   description[32];      /* resource description            */
   unsigned char   reserv3[16];          /* reserved                        */
   unsigned char   nau_address;          /* NAU address                     */
   unsigned char   lu_sscp_sess_active;  /* Is LU-SSCP session active       */
   unsigned char   plu_sess_active;      /* Is PLU-SLU session active       */
   unsigned char   dspu_services;        /* DSPU services                   */
   unsigned char   reserv1;              /* reserved                        */
   SESSION_STATS   lu_sscp_stats;        /* LU-SSCP session statistics      */
   SESSION_STATS   ds_plu_stats;         /* Downstream PLU-SLU session stats */
   SESSION_STATS   us_plu_stats;         /* Upstream PLU-SLU session stats  */
} DOWNSTREAM_LU_INDICATION;
```

```
typedef struct session_stats
{
  AP_UINT16       rcv_ru_size;          /* session receive RU size         */
  AP_UINT16       send_ru_size;         /* session send RU size            */
  AP_UINT16       max_send_btu_size;    /* maximum send BTU size           */
  AP_UINT16       max_rcv_btu_size;     /* maximum rcv BTU size            */
  AP_UINT16       max_send_pac_win;     /* maximum send pacing window size */
  AP_UINT16       cur_send_pac_win;     /* current send pacing window size */
  AP_UINT16       max_rcv_pac_win;      /* maximum receive pacing window size*/
  AP_UINT16       cur_rcv_pac_win;      /* current receive pacing window size*/
  AP_UINT32       send_data_frames;     /* number of data frames sent      */
  AP_UINT32       send_fmd_data_frames; /* num fmd data frames sent        */
  AP_UINT32       send_data_bytes;      /* number of data bytes sent       */
  AP_UINT32       rcv_data_frames;      /* number of data frames received  */
  AP_UINT32       rcv_fmd_data_frames;  /* num fmd data frames received    */
  AP_UINT32       rcv_data_bytes;       /* number of data bytes received   */
  unsigned char   sidh;                 /* session ID high byte (from LFSID) */
  unsigned char   sidl;                 /* session ID low byte (from LFSID)  */
  unsigned char   odai;                 /* ODAI bit set                    */
  unsigned char   ls_name[8];           /* Link station name               */
  unsigned char   reserve;              /* reserved                        */
} SESSION_STATS;
```

パラメーター

オペコード
ワークフロー・ストリームの表示

primary_rc
アプオク

データ損失
以前のダウンストリーム LU の指示が失われたかどうかを指定します。 CS Linux は、標識を送信できない状態 (例えば、内部リソース不足) を検出すると、条件がクリアされた後に次の指示に データ損失パラメーターを設定することによって、これを示します。 可能な値は次のとおりです

類人猿
1 つ以上の直前のダウンストリーム LU 指示が失われました。 この VCB の後のフィールドは、ゼロに設定することができます。

アプ・ノー
以前のダウンストリーム LU 指示は失われませんでした。

dspu_name
ダウンストリーム LU に関連付けられているダウンストリーム PU の名前。 これは、 8 バイトの英数字のタイプ A の EBCDIC ストリング (文字で始まる) で、右側に EBCDIC のスペースが埋め込まれます。

ls_name
ダウンストリーム LU に関連付けられているリンク・ステーションの名前。 これは 8 バイトの ASCII ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

dslu_name
ダウンストリーム LU の名前。 これは、 8 バイトの英数字のタイプ A の EBCDIC ストリング (文字で始まる) で、右側に EBCDIC のスペースが埋め込まれます。

記述
LU の定義に指定されているように、ダウンストリーム LU を記述するヌル終了テキスト・ストリング。

ナウド・アドレス
LU のネットワーク・アクセス可能な装置アドレス。

lu_sscp_sess_active
LU-SSCP セッションがアクティブかどうかを指定します。 可能な値は次のとおりです

類人猿
セッションはアクティブです。

アプ・ノー
セッションはアクティブではありません。

plu_sess_active
PLU-SLU セッションがアクティブかどうかを指定します。 可能な値は次のとおりです

類人猿
セッションはアクティブです。

アプ・ノー
セッションはアクティブではありません。

dspu_services
ローカル・ノードによってダウンストリーム LU に提供されるサービスを指定します。

可能な値は次のとおりです

付加プールの集中
ダウンストリーム LU は SNA ゲートウェイによって提供されます。

アプドラー
ダウンストリーム LU は DLUR によってサービスされます。

LU-SSCP セッションが非活動化されていた場合は、このセッションの session_stats 構造が組み込まれます。 PLU-SLU セッションが非活動化された場合、 session_stats 構造は、ダウンストリームおよびアップストリームの PLU-SLU セッションに組み込まれます。 この構造体のフィールドは以下のとおりです。

rcv_ru_size
最大受信 RU サイズ。 (LU-SSCP セッション統計では、このパラメーターは予約されています。)

send_ru_size
送信 RU の最大サイズ。 (LU-SSCP セッション統計では、このパラメーターは予約されています。)

最大 *send_btu_size*
送信可能な BTU の最大サイズ。

最大 *rcv_btu_size*
受信可能な BTU の最大サイズ。

***send_pac_win* の最大値**
このセッションの「送信ペーシング」ウィンドウの最大サイズ。 (LU-SSCP セッション統計では、このパラメーターは予約されています。)

cur_send_pac_win
このセッションの「送信ペーシング」ウィンドウの現在のサイズ。 (LU-SSCP セッション統計では、このパラメーターは予約されています。)

最大値の *pac_win*
このセッションの受信ペーシング・ウィンドウの最大サイズ。 (LU-SSCP セッション統計では、このパラメーターは予約されています。)

cur_rcv_pac_win
このセッションでの受信ペーシング・ウィンドウの現在のサイズ。 (LU-SSCP セッション統計では、このパラメーターは予約されています。)

send_data_frames
送信された通常フロー・データ・フレームの数。

***send_fmd_data_* フレーム**
送信された通常フロー FMD データ・フレームの数。

send_data_bytes
送信された通常フロー・データ・バイトの数。

rcv_data_frames
受信された通常フロー・データ・フレームの数。

***rcv_fmd_data_* フレーム**
受信された通常フロー FMD データ・フレームの数。

rcv_data_bytes
受信された通常フロー・データ・バイトの数。

シダ
セッション ID の高位バイト。 (アップストリームの PLU-SLU セッション統計では、このパラメーターは予約されています。)

シドル
セッション ID の低位バイト。 (アップストリームの PLU-SLU セッション統計では、このパラメーターは予約されています。)

オーダイ
出荷元宛先の割り当てインディケーター。 セッションを起動すると、ローカル・ノードに 1 次リンク・ステーションが含まれている場合は、 BIND の送信側はこのフィールドをゼロに設定し、 BIND 送信側が 2 次リンク・ステーションを含むノードである場合はそれを 1 つに設定します。 (アップストリームの PLU-SLU セッション統計では、このパラメーターは予約されています。)

ls_name
統計に関連したリンク・ステーション名。 これは 8 バイトの ASCII 文字ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。 (アップストリームの PLU-SLU セッション統計

では、 *dspu_services* が **付加プールの集中**に設定されている場合、このパラメーターは予約されます。)

DOWNSTREAM_PU_INDICATION

This indication is generated when the PU-SSCP session between the downstream PU and the host changes state between active and inactive. When the session becomes inactive, the indication also includes PU-SSCP session statistics.

VCB structure

```
typedef struct downstream_pu_indication
{
 AP_UINT16         opcode;                /* verb operation code           */
 unsigned char     reserv2;               /* reserved                      */
 unsigned char     format;                /* reserved                      */
 AP_UINT16         primary_rc;            /* primary return code           */
 AP_UINT32         secondary_rc;          /* secondary return code         */
 unsigned char     data_lost;             /* previous indication lost      */
 unsigned char     dspu_name[8];          /* PU name                       */
 unsigned char     description[32];       /* resource description          */
 unsigned char     reserv3[16];           /* reserved                      */
 unsigned char     ls_name[8];            /* Link station name             */
 unsigned char     pu_sscp_sess_active;   /* Is PU-SSCP session active     */
 unsigned char     dspu_services;         /* DSPU services                 */
 unsigned char     reserv1[2];            /* reserved                      */
 SESSION_STATS     pu_sscp_stats;         /* PU-SSCP session statistics    */
} DOWNSTREAM_PU_INDICATION;
```

```
typedef struct session_stats
{
 AP_UINT16        rcv_ru_size;          /* session receive RU size           */
 AP_UINT16        send_ru_size;         /* session send RU size              */
 AP_UINT16        max_send_btu_size;    /* maximum send BTU size             */
 AP_UINT16        max_rcv_btu_size;     /* maximum rcv BTU size              */
 AP_UINT16        max_send_pac_win;     /* maximum send pacing window size   */
 AP_UINT16        cur_send_pac_win;     /* current send pacing window size   */
 AP_UINT16        max_rcv_pac_win;      /* maximum receive pacing window size*/
 AP_UINT16        cur_rcv_pac_win;      /* current receive pacing window size*/
 AP_UINT32        send_data_frames;     /* number of data frames sent        */
 AP_UINT32        send_fmd_data_frames; /* num fmd data frames sent          */
 AP_UINT32        send_data_bytes;      /* number of data bytes sent         */
 AP_UINT32        rcv_data_frames;      /* number of data frames received    */
 AP_UINT32        rcv_fmd_data_frames;  /* num fmd data frames received      */
 AP_UINT32        rcv_data_bytes;       /* number of data bytes received     */
 unsigned char    sidh;                 /* session ID high byte (from LFSID) */
 unsigned char    sidl;                 /* session ID low byte (from LFSID)  */
 unsigned char    odai;                 /* ODAI bit set                      */
 unsigned char    ls_name[8];           /* Link station name                 */
 unsigned char    reserve;              /* reserved                          */
} SESSION_STATS;
```

パラメーター

オペコード
ユーザー・ダウン・ストリームの指標

primary_rc
アプオク

データ損失
前のダウンストリーム PU 表示が失われたかどうかを指定します。 CS Linux は、標識を送信できない状態 (例えば、内部リソース不足) を検出すると、条件がクリアされた後に次の指示に データ損失 パラメーターを設定することによって、これを示します。 可能な値は次のとおりです

類人猿
1 つ以上の直前のダウンストリーム PU 表示が失われました。 この VCB の後のフィールドは、ゼロに設定することができます。

アプ・ノー
以前のダウンストリーム PU 表示は失われませんでした。

dspu_name
ダウンストリーム PU の名前。 この名前は 8 バイトの EBCDIC タイプ A ストリングで、名前が 8 文字より短い場合は、右側に EBCDIC のスペースが埋め込まれます。

記述
PU に関連付けられた LS の定義に指定されている、ダウンストリーム PU を記述したヌル終了テキスト・ストリング。

ls_name
ダウンストリーム PU に関連付けられているリンク・ステーションの名前。 これは 8 バイトの ASCII ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

pu_sscp_sess_active
ダウンストリーム PU に対する PU-SSCP セッションが活動状態であるかどうかを指定します。 可能な値は次のとおりです

類人猿
セッションはアクティブです。

アプ・ノー
セッションはアクティブではありません。

dspu_services
ローカル・ノードによってダウンストリーム PU に提供されるサービスを指定します。

可能な値は次のとおりです

付加プールの集中
ダウンストリーム LU は SNA ゲートウェイによって提供されます。

アプドラー
ダウンストリーム LU は DLUR によってサービスされます。

pu_sscp_stats.rcv_ru_size
予約済み (常にゼロに設定されます)。

pu_sscp_stats.send_ru_size
予約済み (常にゼロに設定されます)。

pu_sscp_stats.max_send_btu_size
送信可能な BTU の最大サイズ。

pu_sscp_stats.max_rcv_btu_size
受信可能な BTU の最大サイズ。

pubsscp_stats.max_send_pac_win
予約済み (常にゼロに設定されます)。

pu_sscp_stats.cur_send_pac_win
予約済み (常にゼロに設定されます)。

pu_sscp_stats.max_rcv_pac_win
予約済み (常にゼロに設定されます)。

pu_sscp_stats.cur_rcv_pac_win
予約済み (常にゼロに設定されます)。

pu_sscp_stats.send_data_frames
送信された通常フロー・データ・フレームの数。

pu_sscp_stats.send_fmd_data_フレーム
送信された通常フロー FMD データ・フレームの数。

pu_sscp_stats.send_data_bytes
送信された通常フロー・データ・バイトの数。

pu_sscp_stats.rcv_data_frames
受信された通常フロー・データ・フレームの数。

pu_sscp_stats.rcv_fmd_data_Frame
受信された通常フロー FMD データ・フレームの数。

pu_sscp_stats.rcv_data_bytes
受信された通常フロー・データ・バイトの数。

pu_sscp_stats.sidh
セッション ID の高位バイト。

pu_sscp_stats.sidl
セッション ID の低位バイト。

pu_sscp_stats.o ダイ
出荷元宛先の割り当てインディケーター。 セッションを起動すると、ローカル・ノードに 1 次リンク・ステーションが含まれている場合は、 BIND の送信側はこのフィールドをゼロに設定し、 BIND 送信側が 2 次リンク・ステーションを含むノードである場合はそれを 1 つに設定します。

pu_sscp_stats.ls_name
統計に関連したリンク・ステーション名。 これは 8 バイトの ASCII 文字ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

FOCAL_POINT_INDICATION

This indication is generated whenever a focal point is acquired, changed or revoked.

VCB 構造体

```
typedef struct focal_point_indication
{
  AP_UINT16        opcode;             /* verb operation code               */
  unsigned char    reserv2;            /* reserved                          */
  unsigned char    format;             /* reserved                          */
  AP_UINT16        primary_rc;         /* primary return code               */
  AP_UINT32        secondary_rc;       /* secondary return code             */
  unsigned char    data_lost;          /* previous indication lost          */
  unsigned char    ms_category[8];     /* Focal point category              */
  unsigned char    fp_fqcp_name[17];   /* Fully qualified focal point cp name*/
  unsigned char    ms_appl_name[8];    /* Focal point application name      */
  unsigned char    fp_type;            /* type of current focal point       */
  unsigned char    fp_status;          /* status of focal point             */
  unsigned char    fp_routing;         /* type of MDS routing to reach FP   */
  unsigned char    reserva[20];        /* reserved                          */
} FOCAL_POINT_INDICATION;
```

Parameters

opcode
AP_FOCAL_POINT_INDICATION

primary_rc
AP_OK

data_lost
Specifies whether any previous focal point indications have been lost. If CS Linux detects a condition that prevents it from sending an indication (for example an internal resource shortage), it indicates this by setting the *data_lost* parameter on the next indication after the condition has cleared. Possible values are:

AP_YES
One or more previous focal point indications were lost. Later fields in this VCB may be set to zeros.

AP_NO
No previous focal point indications were lost.

ms_category
Management Services category for which the focal point has changed. This can be either one of the category names specified in the MS Discipline-Specific Application Programs table contained in the *SNA Management Services Reference*, padded on the right with spaces if the name is shorter than 8

bytes, or a user-defined category. User-defined category names should be an 8-byte type-1134 EBCDIC string, padded on the right with spaces if the name is shorter than 8 bytes.

fp_fqcp_name
Fully qualified name of the current focal point for the specified MS category. The name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters. If this parameter is set to 17 binary zeros, this indicates that there is no focal point currently defined for the specified MS category; the previous focal point has been deleted and not replaced.

ms_appl_name
Name of the current focal point application. This can be either one of the application names specified in the MS Discipline-Specific Application Programs table in *Systems Network Architecture: Management Services Reference SC30-3346*, padded on the right with spaces to 8 bytes, or a user-defined application name (see the Bibliography). User-defined names should be an 8-byte type-1134 EBCDIC string, padded on the right with spaces if the name is shorter than 8 bytes. If this parameter is set to 8 binary zeros, this indicates that there is no focal point currently defined for the specified MS category; the previous focal point has been deleted and not replaced.

fp_type
Type of focal point. Refer to SNA Management Services for further detail. Possible values are:

`AP_EXPLICIT_PRIMARY_FP`

`AP_IMPLICIT_PRIMARY_FP`

`AP_BACKUP_FP`

`AP_DEFAULT_PRIMARY_FP`

`AP_DOMAIN_FP`

`AP_HOST_FP`

`AP_NO_FP`

fp_status
Status of the focal point. Possible values are:

AP_NOT_ACTIVE
The focal point has gone from active to inactive.

AP_ACTIVE
The focal point has gone from inactive or pending active to active.

fp_routing
Type of routing that applications should specify when sending data to the focal point. This parameter is used only if the focal point status is AP_ACTIVE. Possible values are:

AP_DEFAULT
Data should be sent using default routing.

AP_DIRECT
Data should be sent using direct routing.

ISR_指標

この指標は、中間セッション・ルーティング (ISR) セッションが活動化または非活動化されると生成されます。 セッションが非活動化されると、戻されたデータには、セッションの使用に関する統計が含まれます

VCB 構造体

```
typedef struct isr_indication
{
  AP_UINT16       opcode;                /* verb operation code            */
  unsigned char   reserv2;               /* reserved                       */
  unsigned char   format;                /* reserved                       */
```

```
  AP_UINT16       primary_rc;             /* primary return code             */
  AP_UINT32       secondary_rc;           /* secondary return code           */
  unsigned char   data_lost;              /* previous indication lost        */
  unsigned char   deactivated;            /* has ISR session been            */
                                          /*  deactivated?                   */
  FQPCID          fqpcid;                 /* FQPCID for ISR session          */
  unsigned char   fqplu_name[17];         /* fully-qualified primary LU name */
  unsigned char   fqslu_name[17];         /* fully-qualified secondary       */
                                          /*  LU name                        */
  unsigned char   mode_name[8];           /* mode name                       */
  unsigned char   cos_name[8];            /* COS name                        */
  unsigned char   transmission_priority;  /* transmission priority           */
  AP_UINT32       sense_data;             /* sense data                      */
  unsigned char   reserv2a[2];            /* reserved                        */
  SESSION_STATS   pri_sess_stats;         /* Primary hop session statistics  */
  SESSION_STATS   sec_sess_stats;         /* Secondary hop session statistics*/
  unsigned char   reserva[20];            /* reserved                        */
} ISR_INDICATION;
```

```
typedef struct fqpcid
{
  unsigned char   pcid[8];              /* procedure correlator identifier   */
  unsigned char   fqcp_name[17];        /* originator's network qualified    */
                                        /*  CP name                          */
  unsigned char   reserve3[3];          /* reserved                          */
} FQPCID;
```

```
typedef struct session_stats
{
  AP_UINT16       rcv_ru_size;          /* session receive RU size           */
  AP_UINT16       send_ru_size;         /* session send RU size              */
  AP_UINT16       max_send_btu_size;    /* maximum send BTU size             */
  AP_UINT16       max_rcv_btu_size;     /* maximum rcv BTU size              */
  AP_UINT16       max_send_pac_win;     /* maximum send pacing window size   */
  AP_UINT16       cur_send_pac_win;     /* current send pacing window size   */
  AP_UINT16       max_rcv_pac_win;      /* maximum receive pacing window size*/
  AP_UINT16       cur_rcv_pac_win;      /* current receive pacing window size*/
  AP_UINT32       send_data_frames;     /* number of data frames sent        */
  AP_UINT32       send_fmd_data_frames; /* num fmd data frames sent          */
  AP_UINT32       send_data_bytes;      /* number of data bytes sent         */
  AP_UINT32       rcv_data_frames;      /* number of data frames received    */
  AP_UINT32       rcv_fmd_data_frames;  /* num fmd data frames received      */
  AP_UINT32       rcv_data_bytes;       /* number of data bytes received     */
  unsigned char   sidh;                 /* session ID high byte (from LFSID) */
  unsigned char   sidl;                 /* session ID low byte (from LFSID)  */
  unsigned char   odai;                 /* ODAI bit set                      */
  unsigned char   ls_name[8];           /* Link station name                 */
  unsigned char   reserve;              /* reserved                          */
} SESSION_STATS;
```

Parameters

opcode
AP_ISR_INDICATION

primary_rc
AP_OK

data_lost
Specifies whether any previous ISR indications have been lost. If CS Linux detects a condition that prevents it from sending an indication (for example an internal resource shortage), it indicates this by setting the *data_lost* parameter on the next indication after the condition has cleared. Possible values are:

AP_YES
One or more previous ISR indications were lost. Later fields in this VCB may be set to zeros.

AP_NO
No previous ISR indications were lost.

deactivated
Specifies whether the ISR session was deactivated or activated. Possible values are:

AP_YES
The session was deactivated.

AP_NO
The session was activated.

fqpcid.pcid
Procedure Correlator ID for the session. This is an 8-byte hexadecimal string.

fqpcid.fqcp_name
Fully qualified control point name. The name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

fqplu_name
Fully qualified name of the primary LU for this session; this parameter is reserved if *deactivated* is set to AP_YES. The name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

fqslu_name
Fully qualified name of the secondary LU for this session; this parameter is reserved if *deactivated* is set to AP_YES. The name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

mode_name
Mode name for this session; this parameter is reserved if *deactivated* is set to AP_YES. This is an 8-byte type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

cos_name
COS name for this session; this parameter is reserved if *deactivated* is set to AP_YES. This is an 8-byte type-A EBCDIC string (starting with a letter), padded to the right with EBCDIC spaces.

transmission_priority
The transmission priority associated with the session. This parameter is reserved if *deactivated* is set to AP_YES.

sense_data
The sense data sent or received on the UNBIND request. This parameter is reserved if *deactivated* is set to AP_NO.

If the ISR session was deactivated, `session_stats` structures are included for the primary and secondary sessions. The fields in this structure are as follows:

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 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.

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 its local node contains the primary link station, and sets it to one if its local node contains the secondary link station.

ls_name
Link station name associated with statistics. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes, which can be used to correlate the session statistics with the link over which the session traffic flows.

ローカル・LU_指標

この指示は、ローカル LU が定義または削除されたときに生成されます。 この指示は、登録済みアプリケーションが、現在定義されているすべてのローカル LU のリストを保守するために使用できます。

VCB structure

```
typedef struct local_lu_indication
{
   AP_UINT16        opcode;             /* Indication operation code        */
   unsigned char    reserv2;            /* reserved                         */
   unsigned char    format;             /* reserved                         */
   AP_UINT16        primary_rc;         /* primary return code              */
   AP_UINT32        secondary_rc;       /* secondary return code            */
   unsigned char    data_lost;          /* Previous indication lost?        */
   unsigned char    reason;             /* reason for this indication       */
   unsigned char    lu_name[8];         /* LU name                          */
   DESCRIPTION      description;        /* resource description             */
   unsigned char    lu_alias[8];        /* LU alias                         */
   unsigned char    nau_address;        /* NAU address                      */
   unsigned char    reserv4;            /* reserved                         */
   unsigned char    pu_name[8];         /* PU name                          */
   unsigned char    lu_sscp_active;     /* Is LU-SSCP session active        */
   unsigned char    reserv5;            /* reserved                         */
   SESSION_STATS    lu_sscp_stats;      /* LU-SSCP session statistics       */
   unsigned char    sscp_id[6];         /* SSCP ID                          */
} LOCAL_LU_INDICATION;
```

```
typedef struct session_stats
{
   AP_UINT16        rcv_ru_size;        /* session receive RU size          */
   AP_UINT16        send_ru_size;       /* session send RU size             */
   AP_UINT16        max_send_btu_size;  /* max send BTU size                */
   AP_UINT16        max_rcv_btu_size;   /* max receive BTU size             */
   AP_UINT16        max_send_pac_win;   /* max send pacing window size      */
   AP_UINT16        cur_send_pac_win;   /* current send pacing window size  */
   AP_UINT16        max_rcv_pac_win;    /* max receive pacing window size   */
```

```
    AP_UINT16        cur_rcv_pac_win;     /* current rcv pacing window size   */
    AP_UINT32        send_data_frames;    /* number of data frames sent       */
    AP_UINT32        send_fmd_data_frames; /* num of fmd data frames sent     */
    AP_UINT32        send_data_bytes;     /* number of data bytes sent        */
    AP_UINT32        rcv_data_frames;     /* number of data frames received   */
    AP_UINT32        rcv_fmd_data_frames; /* num of fmd data frames received  */
    AP_UINT32        rcv_data_bytes;      /* number of data bytes received    */
    unsigned char    sidh;                /* session ID high byte             */
    unsigned char    sidl;                /* session ID low byte              */
    unsigned char    odai;                /*origin-destination assignor bit set*/
    unsigned char    ls_name[8];          /* link station name                */
    unsigned char    pacing_type;         /* type of pacing in use            */
} SESSION_STATS;
```

The LU-SSCP statistics contained in the `session_stats` structure are valid only when both the *nau_address* parameter is set to a nonzero value and the *lu_sscp_active* parameter is set to AP_YES. Otherwise, the parameters in the `session_stats` structure are reserved.

パラメーター

オペコード
ローカル・LU_LU_指標

primary_rc
アプオク

データ損失
前のディレクトリー指示が失われたかどうかを指定します。 CS Linux は、標識を送信できない状態 (例えば、内部リソース不足) を検出すると、条件がクリアされた後に次の指示に データ損失 パラメーターを設定することによって、これを示します。 可能な値は次のとおりです

類人猿
以前の 1 つ以上のディレクトリー指示が失われました。 この VCB の後のフィールドは、ゼロに設定することができます。

アプ・ノー
以前のディレクトリー指示は失われませんでした。

理由
指示の理由。 可能な値は次のとおりです

追加済み
LU が定義されました。

削除済み
DELETE_LOCAL_LU を使用して明示的に、または DELETE_LS、 DELETE_PORT、または DELETE_DLC のいずれかを使用して、 LU が明示的に削除されました。

追加指定がアクティブ
ノードが ACTLU を正常に処理した後で、 LU-SSCP セッションがアクティブになりました。

AP_SSCP_INACTIVE
通常の DACTLU またはリンク障害の後、 LU-SSCP セッションが非活動状態になりました。

lu_name
状態が変更されたローカル論理装置 (LU) の名前。 これは、 8 バイトの英数字のタイプ A の EBCDIC ストリング (文字で始まる) で、右側に EBCDIC のスペースが埋め込まれます。

記述
DEFINE_LOCAL_LU で指定されているリソース記述。

lu_alias
ローカルに定義された LU の別名。 これは、ローカル表示可能文字セットの 8 バイトのストリングです。 8 バイトはすべて意味があります。

ナウド・アドレス
LU のネットワーク・アクセス可能単位 (NAU) アドレス。 この値は、 1-255 の範囲内でなければなりません。 ゼロ以外の値は、 LU が従属 LU であることを意味します。 値 0 (ゼロ) は、 LU が独立 LU であることを意味します。

プール名
この LU が使用する物理装置 (PU) の名前。 これは 8 バイトのタイプ A の EBCDIC ストリングです)。 右側に EBCDIC のスペースが埋め込まれます。 このパラメーターは、 ナウド・アドレス パラメーターが 0 (ゼロ) に設定されていない場合にのみ有効です。 ナウド・アドレス パラメーターが 0 に設定されている場合、 プール名 パラメーターはすべて 2 進ゼロに設定されます。

lu_sscp_sess_active
LU-SSCP セッションがアクティブかどうかを指定します。 ナウド・アドレス パラメーターが 0 (ゼロ) に設定されている場合、このパラメーターは予約されています。 可能な値は次のとおりです

類人猿
LU-SSCP セッションがアクティブです。

アプ・ノー
LU-SSCP セッションがアクティブではありません。

lu_sscp_stats.rcv_ru_size
このパラメーターは常に予約されます。

lu_sscp_stats.send_ru_size
このパラメーターは常に予約されます。

lu_sscp_stats.max_send_btu_size
送信可能な最大基本伝送単位 (BTU) 。

lu_sscp_stats.max_rcv_btu_size
受信できる最大基本伝送単位 (BTU) 。

lu_sscp_stats.max_send_pac_win
このパラメーターは常にゼロに設定されます。

lu_sscp_stats.cur_send_pac_win
このパラメーターは常にゼロに設定されます。

lu_sscp_stats.max_rcv_pac_win
このパラメーターは常にゼロに設定されます。

lu_sscp_stats.cur_rcv_pac_win
このパラメーターは常にゼロに設定されます。

lu_sscp_stats.send_data_Frame
送信された通常フロー・データ・フレームの数。

lu_sscp_stats.send_fmd_data_フレーム
送信された通常フロー機能管理データ (FMD) フレームの数。

lu_sscp_stats.send_data_bytes
送信された通常フロー・データ・バイトの数。

lu_sscp_stats.rcv_data_Frame
受信された通常フロー・データ・フレームの数。

lu_sscp_stats.rcv_fmd_data_Frame
受信された通常フロー機能管理データ (FMD) フレームの数。

lu_sscp_stats.rcv_data_bytes
受信された通常フロー・データ・バイトの数。

lu_sscp_stats.sidh
セッション ID の高位バイト。

lu_sscp_stats.sidl
セッション ID の低位バイト。

lu_sscp_stats.o ダイ
発信元宛先の割り当てインディケーター。 セッションを活動化すると、 ACTLU の送信側は、ローカル・ノードに 1 次リンク・ステーションが含まれている場合はこのパラメーターをゼロに設定し、 ACTLU 送信側が 2 次リンク・ステーションを含んでいるノードである場合はそれを 1 つに設定します。

lu_sscp_stats.ls_name
統計に関連したリンク・ステーション名。 これは、ローカル表示可能文字セットの 8 バイトのストリングです。 8 バイトはすべて意味があります。 このパラメーターを使用すると、このセッションと、セッションが流れるリンクとの関連付けを行うことができます。

lu_sscp_stats.pacing_type
LU-SSCP セッションで使用されている受信ペーシング・タイプ。 これは、 **追加なし**の値を取ります。

sscp_id
この LU によって使用される PU の ACTPU で受信された SSCP の ID 。 このパラメーターは 6 バイトで、従属 LU によってのみ使用されます。 このパラメーターは、独立 LU の場合、または *lu_sscp_sess_active* パラメーターが **類人猿**に設定されていない場合は、すべてゼロに設定されます。

ローカル・トポロジーの指標

この指標は、次のいずれかが発生した場合に生成されます

- ローカル・ノードのトポロジー・データベースの TG は、アクティブと非アクティブの間の状態を変更します。
- ローカル・ノードのトポロジー・データベース内の TG が、静止していて静止していない間の状態を変更する。
- ローカル・ノードのトポロジー・データベース内の TG 上でコンテンション勝者 CP-CP セッションが活動化または非活動化されます。

VCB structure

```
typedef struct local_topology_indication
{
   AP_UINT16       opcode;              /* verb operation code              */
   unsigned char   reserv2;             /* reserved                         */
   unsigned char   format;              /* reserved                         */
   AP_UINT16       primary_rc;          /* primary return code              */
   AP_UINT32       secondary_rc;        /* secondary return code            */
   unsigned char   data_lost;           /* previous indication lost         */
   unsigned char   status;              /* TG status                        */
   unsigned char   dest[17];            /* name of TG destination node      */
   unsigned char   dest_type;           /* TG destination node type         */
   unsigned char   tg_num;              /* TG number                        */
   unsigned char   cp_cp_session_active;   /* CP-CP sessions active?        */
   unsigned char   branch_link_type;       /* Up or down link?              */
   unsigned char   branch_tg;              /* Branch TG?                    */
   unsigned char   reserva[17];            /* reserved                      */
} LOCAL_TOPOLOGY_INDICATION;
```

パラメーター

オペコード
AP_LOCAL_TOPOLOGY_**指標**

primary_rc
アプオク

データ損失
以前のローカル・トポロジー指示が失われたかどうかを指定します。 CS Linux は、標識を送信できない状態 (例えば、内部リソース不足) を検出すると、条件がクリアされた後に次の指示に データ損失 パラメーターを設定することによって、これを示します。 可能な値は次のとおりです

類人猿
以前のローカル・トポロジー指示 (1 つ以上) が失われました。

アプ・ノー
以前のローカル・トポロジー指示は失われませんでした。

状況
TG の状況を指定します。 これは、 **追加なし** または以下の 1 つ以上の値 (論理 **それとも**を使用して結合) にすることができます。

作動可能の状態

TG_CP_CP_SESSIONS セッション

_TG_静止中

デスト
TG の完全修飾宛先ノード名。 この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

dest_type
宛先ノードのタイプ。 可能な値は次のとおりです

ノードの追加 （ _L ）
ノードの終了。

AP_NETWORK_NODE
ネットワーク・ノード。

ファイルの追加
仮想ルーティング・ノード。

サーバー番号
TG に関連付けられている伝送グループ番号。

cp_cp_session_active
ローカル・ノードのコンテンション勝者 CP-CP セッションが活動状態かどうかを指定します。 可能な値は次のとおりです

類人猿
CP-CP セッションは活動状態です。

アプ・ ノー
CP-CP セッションは活動状態ではありません。

不明
CP-CP セッション状況は不明です。

ブランチ・リンク・タイプ
このパラメーターは、ノードが分岐ネットワーク・ノードである場合にのみ適用されます。それ以外の場合は予約済みです

この TG のブランチ・リンク・タイプを指定します。 可能な値は次のとおりです

アプ・ アップリンク
TG はアップリンクです。

AP_DOWNLINK
TG は、エンド・ノードへのダウンリンクです。

リンク・ リンクの追加 （ _BRNN ）
TG は、ローカル・ノードの観点から、エンド・ノードとして表示される分岐ネットワーク・ノードへのダウンリンクです。

アプアザリンク
TG は VRN へのリンクです。

ブランチ・タスク
このパラメーターは、ノードがネットワーク・ノードである場合にのみ適用されます。それ以外の場合は予約済み

TG が分岐 TG であるかどうかを指定します。 可能な値は次のとおりです

類人猿
TG は分岐 TG です。

アプ・ ノー
TG は分岐 TG ではありません。

不明
TG タイプは不明です。

LS_指標

この指示は、リンク・ステーションがアクティブ化または非アクティブ化されたときに生成 リンク・ステーションが非活動化されると、戻りデータにはリンク・ステーションの使用状況に関する統計が含まれます

VCB 構造体

```
typedef struct ls_indication
{
  AP_UINT16       opcode;               /* verb operation code              */
  unsigned char   reserv2;              /* reserved                         */
  unsigned char   format;               /* reserved                         */
  AP_UINT16       primary_rc;           /* primary return code              */
  AP_UINT32       secondary_rc;         /* secondary return code            */
  unsigned char   data_lost;            /* previous indication lost         */
  unsigned char   deactivated;          /* has LS been deactivated?         */
  unsigned char   ls_name[8];           /* link station name                */
  unsigned char   description[32];      /* resource description             */
  unsigned char   reserv1[16];          /* reserved                         */
  unsigned char   adj_cp_name[17];      /* network qualified Adjacent CP name*/
  unsigned char   adj_node_type;        /* adjacent node type               */
  AP_UINT16       act_sess_count;       /* active session count on link     */
  unsigned char   indication_cause;     /* cause of indication              */
  LS_STATS        ls_stats;             /* link station statistics          */
  unsigned char   tg_num;               /* tg number                        */
  AP_UINT32       sense_data;           /* sense data                       */
  unsigned char   brnn_link_type;       /* type of branch link              */
  unsigned char   adj_cp_is_brnn;       /* is adjacent node a BrNN?         */
  unsigned char   mltg_member;          /* reserved                         */
  unsigned char   tg_sharing;           /* reserved                         */
  unsigned char   ls_type;              /* how LS was defined or discovered  */
  unsigned char   reserva[14];          /* reserved                         */
} LS_INDICATION;
```

```
typedef struct ls_stats
{
  AP_UINT32       in_xid_bytes;          /* number of XID bytes received     */
  AP_UINT32       in_msg_bytes;          /* number of message bytes received */
  AP_UINT32       in_xid_frames;         /* number of XID frames received    */
  AP_UINT32       in_msg_frames;         /* number of message frames received*/
  AP_UINT32       out_xid_bytes;         /* number of XID bytes sent         */
  AP_UINT32       out_msg_bytes;         /* number of message bytes sent     */
  AP_UINT32       out_xid_frames;        /* number of XID frames sent        */
  AP_UINT32       out_msg_frames;        /* number of message frames sent    */
  AP_UINT32       in_invalid_sna_frames;        /* number of invalid          */
                                                /*  frames received           */
  AP_UINT32       in_session_control_frames;    /* number of control          */
                                                /*  frames received           */
  AP_UINT32       out_session_control_frames;   /* number of control          */
                                                /*  frames sent               */
  AP_UINT32       echo_rsps;             /* reserved                         */
  AP_UINT32       current_delay;         /* reserved                         */
  AP_UINT32       max_delay;             /* reserved                         */
  AP_UINT32       min_delay;             /* reserved                         */
  AP_UINT32       max_delay_time;        /* reserved                         */
  AP_UINT32       good_xids;             /* successful XID on LS count       */
  AP_UINT32       bad_xids;              /* unsuccessful XID on LS count     */
} LS_STATS;
```

パラメーター

オペコード
追加の指示

primary_rc
アプオク

データ損失
以前の LS 表示が失われたかどうかを指定します。 CS Linux は、標識を送信できない状態 (例えば、内部リソース不足) を検出すると、条件がクリアされた後に次の指示に データ損失 パラメーターを設定することによって、これを示します。 可能な値は次のとおりです

類人猿
1 つ以上の直前の LS 表示が失われました。 この VCB の後のフィールドは、ゼロに設定することができます。

アプ・ノー
以前の LS 指示は失われませんでした。

非活動化
LS が非活動化されたか活動化されたかを指定 可能な値は次のとおりです

類人猿
LS は非活動化されました。

アプ・ノー
LS が活動化されました。

ls_name
リンク・ステーションの名前。 これは 8 バイトの ASCII ストリングで、名前が 8 バイトより短い場合は、右側にスペースが埋め込まれます。

記述
LS を記述するヌル終了のテキスト・ストリング (LS の定義に指定されているもの) 。

付加属性名
隣接ノードの完全修飾 CP 名。 この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

ノード・タイプの調整
隣接ノードのタイプ。 可能な値は次のとおりです

ノードの追加 （ _L ）
ノードの終了。

AP_NETWORK_NODE
ネットワーク・ノード。

AP_LEN_NODE
LEN ノード。

ファイルの追加
仮想ルーティング・ノード。

act_sess_count
リンクを使用したアクティブ・セッション (エンドポイントと中間の両方) の総数。

指示原因
指示の原因。 可能な値は次のとおりです

非アクティブ化
LS が活動化されました。

追加の非アクティブ化が開始されました
LS の非活動化処理が開始されました。

非アクティブ化中
LS は非活動化されました。

AP_SESS_COUNT_変中
LS を使用しているアクティブ・セッションの数が変更されました。

付加 CP_NAME_変更
隣接ノードの CP 名が変更されました。

データが失われた
前の指示を送信できませんでした。

失敗
LS は失敗しました。

追加アクティビティーが開始されました
LS は自動活動化をサポートしており、セッションに必要なときに自動的に開始されます。

追加の活動化が失敗しました
LS は自動活動化をサポートしていますが、必要なときに自動起動を自動的に開始しようとします。

AP_LR_アクティブ化中
LS は失敗した (または活動化しようとして失敗しました) 。 CS Linux はそれを再活動化しようとしています。

以下のパラメーターは、非活動化されて **類人猿**場合にのみ戻されます。これは、 LS が非活動化されていることを

ls_stats.in_xid_bytes
このリンク・ステーションで受信された XID (交換識別) バイトの合計数。

ls_stats.in_msg_bytes
このリンク・ステーションで受信されたデータ・バイトの合計数。

***ls_stats.in_xid_*フレーム**
このリンク・ステーションで受信された XID (交換識別) フレームの合計数。

***ls_stats.in_msg_*フレーム**
このリンク・ステーションで受信されたデータ・フレームの合計数。

ls_stats.out_xid_bytes
このリンク・ステーションで送信された XID (交換識別) バイトの合計数。

ls_stats.out_msg_bytes
このリンク・ステーションで送信されたデータ・バイトの合計数。

***ls_stats.out_xid_*フレーム**
このリンク・ステーションで送信された XID (交換識別) フレームの合計数。

***ls_stats.out_msg_*フレーム**
このリンク・ステーションで送信されたデータ・フレームの合計数。

***ls_stats.in_invalid_sna_*フレーム数**
このリンク・ステーションで受信されなかった SNA フレームの合計数。

***ls_stats.in_session_control_*フレーム**
このリンク・ステーションで受信されたセッション制御フレームの合計数。

***ls_stats.out_session_control_*フレーム**
このリンク・ステーションで送信されたセッション制御フレームの合計数。

ls_stats.good_xid
このリンク・ステーションが開始されてからこのリンク・ステーションで発生した成功した XID 交換の総数。

ls_stats.bad_xids
開始後にこのリンク・ステーションで発生した失敗 XID 交換の合計数。

サーバー番号
LS に関連付けられている伝送グループ番号。

sense_data
LS が XID プロトコル・エラーのために失敗した場合、このパラメーターには、エラーに関連したセンス・データが含まれます。 指示原因 が **失敗**以外の値に設定されている場合、このパラメーターは予約されます。

brnn_link_type
このパラメーターは、ローカル・ノードが分岐ネットワーク・ノードである場合にのみ適用されます。それ以外の場合は予約済み

このリンクのブランチ・リンク・タイプを指定します。 可能な値は次のとおりです

アプ・アップリンク
リンクはアップリンクです。

AP_DOWNLINK
リンクはダウンリンクです。

アプアザリンク
リンクは VRN へのリンクです。

認識されていないリンク・タイプ
ブランチ・リンク・タイプが不明です。

サポートされていない追加の数
このリンクは PU 2.0 トラフィックのみをサポートします。

付属 *cp_is_brnn*
隣接ノードが分岐ネットワーク・ノードであるかどうかを指定します。 可能な値は次のとおりです

類人猿
隣接ノードは分岐ネットワーク・ノードです。

アプ・ノー
隣接ノードは分岐ネットワーク・ノードではありません。

不明
隣接ノード・タイプが不明です。

ls_type
このリンクの定義方法または検出方法を指定します。 可能な値は次のとおりです

定義済みの追加
リンク・ステーションは、 CS Linux 管理プログラムによって明示的に定義されています。

追加 LS_ダイナミック
リンク・ステーションは、ローカル・ノードが接続ネットワークを介して別のノードに接続されたときに作成されたものです。

一時的 （ 一時的 ）
リンク・ステーションは , 着信呼び出しを処理するために一時的に作成されましたが , まだ活動状態になっていません。

暗黙の追加
CS Linux が、定義されたリンク・ステーションと一致しない可能性がある着信呼び出しを受け取ったときに、リンク・ステーションが暗黙的に定義されました。

定義済みの LS_LS_DLUS_DEFINED
リンク・ステーションは、 DLUR から提供されるダウンストリーム PU へのダイナミック・リンク・ステーションであり、ローカル・ノードが DLUS から ACTPU を受信したときに定義されました。

LU_0_TO_3_INDICATION

This indication is generated when the session status of a type 0-3 LU changes.

VCB structure

```
typedef struct lu_0_to_3_indication
{
  AP_UINT16         opcode;                 /* verb operation code         */
  unsigned char     reserv2;                /* reserved                    */
  unsigned char     format;                 /* reserved                    */
  AP_UINT16         primary_rc;             /* primary return code         */
  AP_UINT32         secondary_rc;           /* secondary return code       */
  unsigned char     data_lost;              /* previous indication lost    */
  unsigned char     pu_name[8];             /* PU Name                     */
  unsigned char     lu_name[8];             /* LU Name                     */
  unsigned char     description[32];        /* resource description        */
  unsigned char     reserv1[16];            /* reserved                    */
  unsigned char     nau_address;            /* NAU address                 */
  unsigned char     lu_sscp_sess_active;    /* Is SSCP session active?     */
```

```
  unsigned char     appl_conn_active;         /* Is application using LU?    */
  unsigned char     plu_sess_active;          /* Is PLU-SLU session active?  */
  unsigned char     host_attachment;          /* Host attachment             */
  SESSION_STATS     lu_sscp_stats;            /* LU-SSCP session statistics  */
  SESSION_STATS     plu_stats;                /* PLU session statistics      */
  unsigned char     sscp_id[6];               /* SSCP id                     */
} LU_0_TO_3_INDICATION;
```

```
typedef struct session_stats
{
  AP_UINT16        rcv_ru_size;          /* session receive RU size          */
  AP_UINT16        send_ru_size;         /* session send RU size             */
  AP_UINT16        max_send_btu_size;    /* maximum send BTU size            */
  AP_UINT16        max_rcv_btu_size;     /* maximum rcv BTU size             */
  AP_UINT16        max_send_pac_win;     /* maximum send pacing window size  */
  AP_UINT16        cur_send_pac_win;     /* current send pacing window size  */
  AP_UINT16        max_rcv_pac_win;      /* maximum receive pacing window    */
                                         /*  size                            */
  AP_UINT16        cur_rcv_pac_win;      /* current receive pacing window    */
                                         /*  size                            */
  AP_UINT32        send_data_frames;     /* number of data frames sent       */
  AP_UINT32        send_fmd_data_frames; /* num fmd data frames sent         */
  AP_UINT32        send_data_bytes;      /* number of data bytes sent        */
  AP_UINT32        rcv_data_frames;      /* number of data frames received   */
  AP_UINT32        rcv_fmd_data_frames;  /* num fmd data frames received     */
  AP_UINT32        rcv_data_bytes;       /* number of data bytes received    */
  unsigned char    sidh;                 /* session ID high byte (from LFSID)*/
  unsigned char    sidl;                 /* session ID low byte (from LFSID) */
  unsigned char    odai;                 /* ODAI bit set                     */
  unsigned char    ls_name[8];           /* Link station name                */
  unsigned char    reserve;              /* reserved                         */
} SESSION_STATS;
```

パラメーター

オペコード
追加 0 から 3 への指示

primary_rc
アプオク

データ損失
以前の LU 0 から 3 の指示が失われたかどうかを指定します。 CS Linux は、標識を送信できない状態 (例えば、内部リソース不足) を検出すると、条件がクリアされた後に次の指示に データ損失 パラメーターを設定することによって、これを示します。 可能な値は次のとおりです

類人猿
以前の 1 つ以上の LU 0-3 の指示が失われました。 この VCB の後のフィールドは、ゼロに設定することができます。

アプ・ノー
以前の LU 0-3 の指示は失われませんでした。

プール名
LU が使用するローカル PU の名前。 これは 8 バイトのタイプ A の EBCDIC ストリング (文字で始まる) で、名前が 8 文字より短い場合は、右側に EBCDIC のスペースが埋め込まれます。

lu_name
セッション状況が変更された LU の名前。 これは 8 バイトのタイプ A の EBCDIC ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

記述
LU を記述するヌル終了のテキスト・ストリング (LU の定義に指定されているもの) 。

ナウド・アドレス
LU のネットワーク・アクセス可能な装置アドレス。

lu_sscp_sess_active
SSCP セッションが活動状態であるかどうか (つまり、 ACTLU が正常に処理されたかどうか) を指定します。 可能な値は次のとおりです

類人猿
セッションはアクティブです。

アプ・ノー
セッションはアクティブではありません。

アクティブのアプリケーション接続
アプリケーションが LU を使用しているかどうかを示します 可能な値は次のとおりです

類人猿
アプリケーションが LU を使用しています。

アプ・ノー
LU を使用するアプリケーションがありません。

plu_sess_active
PLU-SLU セッションが活動化されたかどうかを指定します。 可能な値は次のとおりです

類人猿
セッションはアクティブです。

アプ・ノー
セッションはアクティブではありません。

ホスト接続
LU ホスト接続機構タイプ。

可能な値は次のとおりです

追加指示が付加されます
LU は、ホスト・システムに直接接続されます。

付加された付加
LU は DLUR を使用してホスト・システムに接続される

sscp_id
従属 LU セッションの場合、このパラメーターは、ローカル LU がマップされている PU のホストから、 ACTPU で受信された SSCP ID です。 独立 LU セッションの場合、このパラメーターは 0 (ゼロ) に設定されます。 この値は、 16 進値として表示される 6 バイトの配列です。

2 つのセッション (LU-SSCP セッションおよび PLU-SLU セッション) のそれぞれについて、 `session_stats` 構造が組み込まれています。 セッションがアクティブから非アクティブになる場合、構造には以下のパラメーターが含まれます。それ以外の場合、これらのパラメーターは予約され

rcv_ru_size
最大受信 RU サイズ。 (LU-SSCP セッション統計では、このパラメーターは予約されています。)

send_ru_size
送信 RU の最大サイズ。 (LU-SSCP セッション統計では、このパラメーターは予約されています。)

最大 *send_btu_size*
送信可能な BTU の最大サイズ。

最大 *rcv_btu_size*
受信可能な BTU の最大サイズ。

***send_pac_win* の最大値**
このセッションの「送信ペーシング」ウィンドウの最大サイズ。 (LU-SSCP セッション統計では、このパラメーターは予約されています。)

cur_send_pac_win
このセッションの「送信ペーシング」ウィンドウの現在のサイズ。 (LU-SSCP セッション統計では、このパラメーターは予約されています。)

最大値の *pac_win*
このセッションの受信ペーシング・ウィンドウの最大サイズ。 (LU-SSCP セッション統計では、このパラメーターは予約されています。)

cur_rcv_pac_win
このセッションでの受信ペーシング・ウィンドウの現在のサイズ。 (LU-SSCP セッション統計では、このパラメーターは予約されています。)

send_data_frames
送信された通常フロー・データ・フレームの数。

send_fmd_data_フレーム
送信された通常フロー FMD データ・フレームの数。

send_data_bytes
送信された通常フロー・データ・バイトの数。

rcv_data_frames
受信された通常フロー・データ・フレームの数。

rcv_fmd_data_フレーム
受信された通常フロー FMD データ・フレームの数。

rcv_data_bytes
受信された通常フロー・データ・バイトの数。

シダ
セッション ID の高位バイト。

シドル
セッション ID の低位バイト。

オーダイ
出荷元宛先の割り当てインディケーター。 セッションを起動すると、ローカル・ノードに 1 次リンク・ステーションが含まれている場合は、 BIND の送信側はこのフィールドをゼロに設定し、 BIND 送信側が 2 次リンク・ステーションを含むノードである場合はそれを 1 つに設定します。

ls_name
統計に関連したリンク・ステーション名。 これは 8 バイトの ASCII 文字ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

MODE_INDICATION

This indication is sent when a local LU and partner LU start to communicate using a particular mode, or when the active session count for the LU-LU-mode combination changes.

VCB 構造体

```
typedef struct mode_兆候 { AP_UINT16 opcode; /* verb operation code */ unsigned char
reserv2; /* reserved */ unsigned char format; /* reserved */ AP_UINT16 primary_rc; /* primary
return code */ AP_UINT32 secondary_rc; /* secondary return code */ unsigned char data_lost; /*
previous indication lost */ unsigned char removed; /* is entry being removed? */ unsigned char
lu_alias[8]; /* LU alias */ unsigned char plu_alias[8]; /* partner LU alias */ unsigned char
fqplu_name[17]; /* fully qualified partner LU name */ unsigned char mode_name[8]; /* mode name
*/ unsigned char description[32]; /* resource description */ unsigned char reserv1[16]; /*
reserved */ AP_UINT16 curr_sess_count; /* current session count */ unsigned char
reserva[20]; /* reserved */ } MODE_INDICATION;
```

Parameters

opcode
AP_MODE_INDICATION

primary_rc
AP_OK

data_lost
Specifies whether any previous mode indications have been lost. If CS Linux detects a condition that prevents it from sending an indication (for example an internal resource shortage), it indicates this by setting the *data_lost* parameter on the next indication after the condition has cleared. Possible values are:

AP_YES
One or more previous mode indications were lost.

AP_NO
No previous mode indications were lost.

removed
This parameter is currently not used; a mode indication is generated only when the LUs start to use the mode, and not when they stop using it.

lu_alias
Locally defined LU alias. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes.

plu_alias
Partner LU alias. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes.

fqplu_name
Fully qualified name of the partner LU. The name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

mode_name
Mode name which designates the network properties for a group of sessions. This is an 8-byte type-A EBCDIC string (starting with a letter), padded on the right with spaces if the name is shorter than 8 characters.

description
A null-terminated text string describing the mode, as specified in the definition of the mode.

curr_sess_count
The number of sessions currently active for this LU-LU-mode combination.

NN_TOPOLOGY_NODE_指標

この指示は、ネットワーク・ノードのトポロジー・データベース内のノード項目が活動化または非活動化されるときに生成されます

VCB structure

```
typedef struct nn_topology_node_indication
{
    AP_UINT16        opcode;              /* verb operation code            */
    unsigned char    reserv2;             /* reserved                       */
    unsigned char    format;              /* reserved                       */
    AP_UINT16        primary_rc;          /* primary return code            */
    AP_UINT32        secondary_rc;        /* secondary return code          */
    unsigned char    data_lost;           /* previous indication lost       */
    unsigned char    deactivated;         /* has the node become inactive?  */
    unsigned char    node_name[17];       /* node name                      */
    unsigned char    node_type;           /* node type                      */
    unsigned char    branch_aware;        /* is the node branch aware?      */
    unsigned char    reserva[19];         /* reserved                       */
} NN_TOPOLOGY_NODE_INDICATION;
```

Parameters

opcode
AP_NN_TOPOLOGY_NODE_INDICATION

primary_rc
AP_OK

data_lost
Specifies whether any previous NN topology node indications have been lost. If CS Linux detects a condition that prevents it from sending an indication (for example an internal resource shortage), it

indicates this by setting the *data_lost* parameter on the next indication after the condition has cleared. Possible values are:

AP_YES
One or more previous NN topology node indications were lost. Later fields in this VCB may be set to zeros.

AP_NO
No previous NN topology node indications were lost.

deactivated
Specifies whether the node has been deactivated or activated. Possible values are:

AP_YES
The node has been deactivated.

AP_NO
The node has been activated.

node_name
Network qualified node name from Network Topology Database. The name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

node_type
Type of the node. Possible values are:

AP_NETWORK_NODE
Network node.

AP_VRN
Virtual routing node.

branch_aware
Specifies whether the node supports branch awareness, APPN Option Set 1120.

AP_NO
The node does not support option set 1120.

AP_YES
The node supports option set 1120.

NN_TOPOLOGY_TG_INDICATION

This indication is generated when a TG entry in a network node's topology database is activated or deactivated.

VCB structure

```
typedef struct nn_topology_tg_indication
{
    AP_UINT16        opcode;               /* verb operation code            */
    unsigned char    reserv2;              /* reserved                       */
    unsigned char    format;               /* reserved                       */
    AP_UINT16        primary_rc;           /* primary return code            */
    AP_UINT32        secondary_rc;         /* secondary return code          */
    unsigned char    data_lost;            /* previous indication lost       */
    unsigned char    status;               /* TG status                      */
    unsigned char    owner[17];            /* name of TG owner node          */
    unsigned char    dest[17];             /* name of TG destination node    */
    unsigned char    tg_num;               /* TG number                      */
    unsigned char    owner_type;           /* type of node that owns TG      */
    unsigned char    dest_type;            /* TG destination node type       */
    unsigned char    cp_cp_session_active; /* are CP-CP sessions active?     */
    unsigned char    branch_tg;            /* is this a branch link?         */
    unsigned char    multilink_tg;         /* reserved                       */
    unsigned char    reserva[15];          /* reserved                       */
} NN_TOPOLOGY_TG_INDICATION;
```

パラメーター

オペコード
トポロジー・トポロジーの追加の指標

primary_rc
アプオク

データ損失
以前の NN トポロジー TG 指示が失われたかどうかを示します。 CS Linux は、標識を送信できない状態 (例えば、内部リソース不足) を検出すると、条件がクリアされた後に次の指示に データ損失 パラメーターを設定することによって、これを示します。 可能な値は次のとおりです

類人猿
以前の NN トポロジー TG 指示 (1 つ以上) が失われました。 この VCB の後のフィールドは、ゼロに設定することができます。

アプ・ノー
以前の NN トポロジー TG 指示は失われませんでした。

状況
TG の状況を指定します。 これは、 **追加なし**、または以下の 1 つ以上の値 (論理 **それとも**を使用して結合) にすることができます。

作動可能の状態

TG_CP_CP_SESSIONS セッション

_TG_静止中

所有者
TG の発信元ノード (CS Linux ローカル・ノード名) の名前。 この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

デスト
TG の完全修飾宛先ノード名。 この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。 これは、最大 8 つの A ストリング文字のネットワーク ID 、 EBCDIC ドット (ピリオド) 文字、および最大 8 文字の A ストリング文字のネットワーク名で構成されます。

サーバー番号
TG に関連付けられている伝送グループ番号。

所有者タイプ
TG を所有するノードのタイプ。 可能な値は次のとおりです

AP_NETWORK_NODE

ファイルの追加

dest_type
TG の宛先ノードのタイプ。 可能な値は次のとおりです

AP_NETWORK_NODE

ファイルの追加

cp_cp_session_active
所有ノードの競合勝者 CP-CP セッションがアクティブであるかどうかを指定します。 可能な値は次のとおりです

類人猿
CP-CP セッションは活動状態です。

アプ・ノー
CP-CP セッションは活動状態ではありません。

不明
CP-CP セッション状況は不明です。

ブランチ・タスク
TG が分岐 TG であるかどうかを指定します。 可能な値は次のとおりです

類人猿
TG は分岐 TG です。

アプ・ノー
TG は分岐 TG ではありません。

不明
TG タイプは不明です。

NOF_STATUS_INDICATION

This indication is generated when the application can no longer access its connected target (because the CS Linux software on the target computer has been stopped, or because the communications path to the target computer has failed). If the target is the domain configuration file, it is also generated if another server takes over as controller (and therefore the connected target file is no longer the controlling copy of the file).

The application does not need to register explicitly for this indication. CS Linux will return it to any application that has registered for any type of indications on the specified target handle. If the application is currently registered to receive indications using more than one callback routine, CS Linux returns this indication to the first routine registered.

After the application receives an indication that the target can no longer be accessed, all subsequent verbs using the relevant target handle will be rejected, apart from DISCONNECT_NODE or CLOSE_FILE (to end the application's connection to the target). In addition, any registrations for indications on this target handle will be lost; in order to continue receiving indications when the target becomes available, the application must reconnect to the target and reregister for the required indications.

VCB 構造体

```
typedef struct nof_status_indication
{
   AP_UINT16        opcode;                /* verb operation code          */
   unsigned char    reserv2;               /* reserved                     */
   unsigned char    format;                /* reserved                     */
   AP_UINT16        primary_rc;            /* primary return code          */
   AP_UINT32        secondary_rc;          /* secondary return code        */
   AP_UINT32        status;                /* status being reported        */
   AP_UINT32        dead_target_handle;    /* Handle of dead connection    */
                                           /*  NULL for system termination */
   unsigned char    reserv1[32];           /* reserved                     */
} NOF_STATUS_INDICATION;
```

Parameters

opcode
AP_NOF_STATUS_INDICATION

primary_rc
AP_OK

status
Specifies the status change being reported. Possible values are:

AP_LOCAL_ABENDED
The CS Linux software on the local computer has stopped. The application should not attempt to issue any more NOF verbs until the software has been restarted.

AP_TARGET_ABENDED
The CS Linux software on the target computer has stopped or the communications path to it has failed.

AP_CONTROLLER_TAKEOVER
This value is returned only when the application is connected to the controlling configuration file (specified by the *requested_role* parameter on OPEN_FILE). Another server has now taken over as controller, so the target file is no longer the controlling configuration file. If the application needs to make further changes to the running configuration, it must use CLOSE_FILE to end its connection with the file, and then issue OPEN_FILE again to access the new controlling configuration file.

dead_target_handle
The target handle of the failed target or of the file that is no longer the controlling configuration file. The application should not attempt to issue any further verbs for this target handle except DISCONNECT_NODE or CLOSE_FILE.

If *status* is set to AP_LOCAL_ABENDED, this parameter is reserved.

PLU_INDICATION

This indication is generated when a local LU begins to communicate with a partner LU. This occurs either when the first ALLOCATE to this PLU is processed or when the first BIND is received from this PLU. The indication is also generated if the partner LU's CP name changes.

VCB structure

```
typedef struct plu_indication
{
   AP_UINT16        opcode;                 /* verb operation code        */
   unsigned char    reserv2;                /* reserved                   */
   unsigned char    format;                 /* reserved                   */
   AP_UINT16        primary_rc;             /* primary return code        */
   AP_UINT32        secondary_rc;           /* secondary return code      */
   unsigned char    data_lost;              /* has previous indication    */
                                            /*  been lost?                */
   unsigned char    removed;                /* is entry being removed?    */
   unsigned char    lu_alias[8];            /* LU alias                   */
   unsigned char    plu_alias[8];           /* partner LU alias           */
   unsigned char    fqplu_name[17];         /* fully qualified partner    */
                                            /*  LU name                   */
   unsigned char    description[32];        /* resource description       */
   unsigned char    reserv1[16];            /* reserved                   */
   unsigned char    partner_cp_name[17];    /* partner CP name            */
   unsigned char    partner_lu_located;     /* partner CP name resolved?  */
   unsigned char    reserva[20];            /* reserved                   */
} PLU_INDICATION;
```

Parameters

opcode
AP_PLU_INDICATION

primary_rc
AP_OK

data_lost
Specifies whether any previous PLU indications have been lost. If CS Linux detects a condition that prevents it from sending an indication (for example an internal resource shortage), it indicates this by setting the *data_lost* parameter on the next indication after the condition has cleared. Possible values are:

AP_YES
One or more previous PLU indications were lost. Later fields in this VCB may be set to zeros.

AP_NO
No previous PLU indications were lost.

removed
This parameter is currently not used; a PLU indication is generated only when the LUs start to communicate, and not when they stop communicating.

lu_alias
Local LU alias. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes.

plu_alias
Partner LU alias. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes.

fqplu_name
17-byte fully qualified network name for the partner LU. The name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

description
A null-terminated text string describing the partner LU, as specified in the definition of the partner LU.

partner_cp_name
17-byte fully qualified network name for the CP associated with the partner LU. This parameter is not used if *partner_lu_located* below is set to AP_NO.

The name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

partner_lu_located
Specifies whether the local node has located the CP where the partner LU is located. Possible values are:

AP_YES
The partner LU has been located. The *partner_cp_name* parameter contains the CP name of the partner LU.

AP_NO
The partner LU has not yet been located. The *partner_cp_name* parameter should not be checked.

ポート指示

この指示は、ポートがアクティブ化または非アクティブ化されたときに生成さ

VCB 構造体

```
typedef struct port_indication
{
   AP_UINT16        opcode;               /* verb operation code           */
   unsigned char    reserv2;              /* reserved                      */
   unsigned char    format;               /* reserved                      */
   AP_UINT16        primary_rc;           /* primary return code           */
   AP_UINT32        secondary_rc;         /* secondary return code         */
   unsigned char    data_lost;            /* previous indication lost      */
   unsigned char    deactivated;          /* has session been deactivated? */
   unsigned char    port_name[8];         /* port name                     */
   unsigned char    description[32];      /* resource description          */
   unsigned char    reserv1[16];          /* reserved                      */
   unsigned char    reserva[20];          /* reserved                      */
} PORT_INDICATION;
```

Parameters

opcode
AP_PORT_INDICATION

primary_rc
AP_OK

data_lost
Specifies whether any previous port indications have been lost. If CS Linux detects a condition that prevents it from sending an indication (for example an internal resource shortage), it indicates this by

setting the *data_lost* parameter on the next indication after the condition has cleared. Possible values are:

AP_YES
One or more previous port indications were lost. Later fields in this VCB may be set to zeros.

AP_NO
No previous port indications were lost.

deactivated
Specifies whether the port has been deactivated or activated. Possible values are:

AP_YES
The port has been deactivated.

AP_NO
The port has been activated.

port_name
Name of port. This is an 8-byte ASCII string, padded on the right with spaces if the name is shorter than 8 bytes.

description
A null-terminated text string describing the port, as specified in the definition of the port.

書き込みの表示

この指示は、ローカル PU の PU-SSCP セッション状況が変更された場合に生成されます。

VCB structure

```
typedef struct pu_indication
{
   AP_UINT16         opcode;                 /* verb operation code        */
   unsigned char     reserv2;                /* reserved                   */
   unsigned char     format;                 /* reserved                   */
   AP_UINT16         primary_rc;             /* primary return code        */
   AP_UINT32         secondary_rc;           /* secondary return code      */
   unsigned char     data_lost;              /* previous indication lost   */
   unsigned char     pu_name[8];             /* PU Name                    */
   unsigned char     description[32];        /* resource description       */
   unsigned char     reserv3[16];            /* reserved                   */
   unsigned char     pu_sscp_sess_active;    /* Is SSCP session active?    */
   unsigned char     host_attachment;        /* Host attachment            */
   unsigned char     reserv1[2];             /* reserved                   */
   SESSION_STATS     pu_sscp_stats;          /* PU-SSCP session statistics */
   unsigned char     sscp_id[6];             /* SSCP id                    */
} PU_INDICATION;
```

```
typedef struct session_stats
{
  AP_UINT16        rcv_ru_size;           /* session receive RU size         */
  AP_UINT16        send_ru_size;          /* session send RU size            */
  AP_UINT16        max_send_btu_size;     /* maximum send BTU size           */
  AP_UINT16        max_rcv_btu_size;      /* maximum rcv BTU size            */
  AP_UINT16        max_send_pac_win;      /* maximum send pacing window size */
  AP_UINT16        cur_send_pac_win;      /* current send pacing window size */
  AP_UINT16        max_rcv_pac_win;       /* maximum receive pacing          */
                                          /*  window size                    */
  AP_UINT16        cur_rcv_pac_win;       /* current receive pacing          */
                                          /*  window size                    */
  AP_UINT32        send_data_frames;      /* number of data frames sent      */
  AP_UINT32        send_fmd_data_frames;  /* num fmd data frames sent        */
  AP_UINT32        send_data_bytes;       /* number of data bytes sent       */
  AP_UINT32        rcv_data_frames;       /* number of data frames received  */
  AP_UINT32        rcv_fmd_data_frames;   /* num fmd data frames received    */
  AP_UINT32        rcv_data_bytes;        /* number of data bytes received   */
  unsigned char    sidh;                  /* session ID high byte            */
                                          /*  (from LFSID)                   */
  unsigned char    sidl;                  /* session ID low byte (from LFSID)*/
  unsigned char    odai;                  /* ODAI bit set                    */
  unsigned char    ls_name[8];            /* Link station name               */
```

```
  unsigned char    reserve;              /* reserved                        */
} SESSION_STATS;
```

パラメーター

オペコード
付加の指標

primary_rc
アプオク

データ損失
以前の PU 表示が失われたかどうかを指定します。 CS Linux は、標識を送信できない状態 (例えば、内部リソース不足) を検出すると、条件がクリアされた後に次の指示に データ損失 パラメーターを設定することによって、これを示します。 可能な値は次のとおりです

類人猿
1 つ以上の以前の PU 表示が失われました。 この VCB の後のフィールドは、ゼロに設定することができます。

アプ・ノー
以前の PU 表示は失われませんでした。

プール名
PU の名前 (DEFINE_LS verb で指定します) 。 これは 8 バイトのタイプ A の EBCDIC ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

記述
PU を記述したヌルで終了するテキスト・ストリング (PU の定義に指定されているもの) 。

pu_sscp_sess_active
PU-SSCP セッションが活動状態であるかどうか (ACTPU が正常に処理されたかどうか) を指定します。 可能な値は次のとおりです

類人猿
PU-SSCP セッションがアクティブです。

アプ・ノー
PU-SSCP セッションが非アクティブです。

ホスト接続
ローカル PU ホスト接続機構タイプ。

可能な値は次のとおりです

追加指示が付加されます
PU はホスト・システムに直接接続されます。

付加された付加
PU は DLUR からサポートされます。

sscp_id
従属 LU セッションの場合、このパラメーターは、ローカル LU がマップされている PU のホストから、 ACTPU で受信された SSCP ID です。 独立 LU セッションの場合、このパラメーターは 0 (ゼロ) に設定されます。 この値は、 16 進値として表示される 6 バイトの配列です。

以下のパラメーターは、セッション状態がアクティブから非アクティブに変更された場合にのみ使用されます。

pu_sscp_stats.rcv_ru_size
予約済み (常にゼロに設定されます)。

pu_sscp_stats.send_ru_size
予約済み (常にゼロに設定されます)。

pu_sscp_stats.max_send_btu_size
送信可能な BTU の最大サイズ。

pu_sscp_stats.max_rcv_btu_size
受信可能な BTU の最大サイズ。

pubsscp_stats.max_send_pac_win
予約済み (常にゼロに設定されます)。

pu_sscp_stats.cur_send_pac_win
予約済み (常にゼロに設定されます)。

pu_sscp_stats.max_rcv_pac_win
予約済み (常にゼロに設定されます)。

pu_sscp_stats.cur_rcv_pac_win
予約済み (常にゼロに設定されます)。

pu_sscp_stats.send_data_frames
送信された通常フロー・データ・フレームの数。

pu_sscp_stats.send_fmd_data_フレーム
送信された通常フロー FMD データ・フレームの数。

pu_sscp_stats.send_data_bytes
送信された通常フロー・データ・バイトの数。

pu_sscp_stats.rcv_data_frames
受信された通常フロー・データ・フレームの数。

pu_sscp_stats.rcv_fmd_data_Frame
受信された通常フロー FMD データ・フレームの数。

pu_sscp_stats.rcv_data_bytes
受信された通常フロー・データ・バイトの数。

pu_sscp_stats.sidh
セッション ID の高位バイト。

pu_sscp_stats.sidl
セッション ID の低位バイト。

pu_sscp_stats.o ダイ
出荷元宛先の割り当てインディケーター。 セッションを起動すると、ローカル・ノードに 1 次リンク・ステーションが含まれている場合は、 BIND の送信側はこのフィールドをゼロに設定し、 BIND 送信側が 2 次リンク・ステーションを含むノードである場合はそれを 1 つに設定します。

pu_sscp_stats.ls_name
統計に関連したリンク・ステーション名。 これは 8 バイトの ASCII 文字ストリングで、名前が 8 文字より短い場合は、右側にスペースが埋め込まれます。

RAPI_CLIENT_指標

この指示は、リモート API クライアントが CS Linux サーバーに接続するか、または CS Linux サーバーから切断したときに生成されます。 NOF アプリケーションは、これらの指示を使用して、どのクライアントが現在そのサーバーをコントローラー・サーバーとして使用しているかを追跡することができます。

VCB structure

```
typedef struct rapi_client_indication
{
   AP_UINT16        opcode;                 /* verb operation code          */
   unsigned char    reserv2;                /* reserved                     */
   unsigned char    format;                 /* reserved                     */
   AP_UINT16        primary_rc;             /* primary return code          */
   AP_UINT32        secondary_rc;           /* secondary return code        */
   unsigned char    data_lost;              /* previous indication lost     */
   unsigned char    reason;                 /* reason for indication        */
   unsigned char    sys_name[128];          /* system name client sends us  */
   SNA_IP_ADDR      rapi_client_origin_ip_addr; /* IP addr client sends us  */
   SNA_IP_ADDR      rapi_client_adj_ip_addr; /* IP addr client comes in on  */
   AP_UINT16        rapi_client_adj_port;   /* port IP client comes in on   */
```

```
   unsigned char    reserva[16];             /* reserved                    */
} RAPI_CLIENT_INDICATION;
```

```
typedef struct sna_ip_addr
{
   AP_UINT16        family;                   /* IPv4 or IPv6                */
   union
   {
     unsigned char  ipv4_addr[4];
     unsigned char  ipv6_addr[16];
   } ip_addr;
} SNA_IP_ADDR;
```

パラメーター

オペコード
アプリケーションでの通知の表示

primary_rc
アプオク

データ損失
以前のクライアント指示が失われたかどうかを指定します。 CS Linux は、標識を送信できない状態 (例えば、内部リソース不足) を検出すると、条件がクリアされた後に次の指示に データ損失 パラメーターを設定することによって、これを示します。 可能な値は次のとおりです

類人猿
以前の 1 つ以上のクライアント指示が失われました。 この VCB の後のフィールドは、ゼロに設定することができます。

アプ・ノー
以前のクライアント指示は失われませんでした。

理由
このクライアントに対して発生した状況変更を指定します。 可能な値は次のとおりです

AP_RAPI_CLIENT_接続済み
クライアントが始動し、この CS Linux サーバーにコントローラー・サーバーとして接続されています。

付加が切断されている
クライアントが停止し、 CS Linux サーバーから切断されました。

システム名
クライアントの完全修飾システム名 (newbox.this.co.uk など) 。

クライアントの発信元 *_ip_addr*
クライアントの IP アドレス。

***client_origin_ip_addr.*家系**
クライアントに指定された TCP/IP アドレスのタイプ。 使用可能な値は以下のとおりです。 (これらは、 CS Linux によって定義される **追加 *** 値ではなく、標準 TCP/IP 値です。)

アフリネット
IPv4 アドレス。小数点付き 10 進数アドレス (193.1.11.100 など) として指定されます。

ネット受信 6
IPv6 アドレス。コロン付き 16 進アドレス (2001:0db8:0000:0000:0000:0000:0000:1428:57ab や 2001:db8::1428:57ab など) として指定します。

注 : 値 **アフリネット** と **ネット受信 6** は、システム・ヘッダー・ファイルから取得され、 CS Linux によって定義された標準の **追加 *** 値ではありません。 システム・ヘッダー・ファイルは /usr/include/linux/socket.h は、 Linux サーバーまたはクライアント上で、 /usr/include/sys/socket.h は AIX クライアント上にあります。 です。

If your NOF application needs to test against these values, you should use #include to include this system file in addition to the **ノード・キュー・ファイル** ヘッダー・ファイル。

アプリケーション・アドレスがレイプされました。 *ip_addr.ipv4_addr*
このフィールドは、 家族 パラメーターが **アフリネット**に設定されている場合にのみ使用されます。 クライアント・コンピューターの IPv4 (ドット 10 進) アドレス。

ユーザーの *ip_addr.ip_addr.ipv6_addr* にレイプされました。
このフィールドは、 家族 パラメーターが **ネット受信 6** に設定されている場合にのみ使用されます。 クライアント・コンピューターの IPv6 (コロン 16 進) アドレス。

アドレス・アドレスのレイプ *_adj_addr*
クライアントが CS Linux に接続するときに使用する IP アドレス。 これは、以下のいずれかが真の場合には クライアントの発信元 *_ip_addr* と同じではありません。

- クライアントは Web サーバーを介して接続します。
- クライアントは、 TCP/IP プロキシーまたは NAT ルーター(Linux iptables ツールなど) を介して接続します。
- クライアントに複数の IP アドレスがあります。

アドレス・アドレスのレイプ
クライアントが CS Linux に接続するときに使用する TCP/IP アドレスのタイプ。 使用可能な値は以下のとおりです。 (これらは、 CS Linux によって定義される **追加 *** 値ではなく、標準 TCP/IP 値です。)

アフリネット
IPv4 アドレス。小数点付き 10 進数アドレス (`193.1.11.100` など) として指定されます。

ネット受信 6
IPv6 アドレス。コロン付き 16 進アドレス (`2001:0db8:0000:0000:0000:0000:0000:1428:57ab` や `2001:db8::1428:57ab` など) として指定します。

注 : 値 **アフリネット** と **ネット受信 6** は、システム・ヘッダー・ファイルから取得され、 CS Linux によって定義された標準の **追加 *** 値ではありません。 システム・ヘッダー・ファイルは `/usr/include/linux/socket.h` は、 Linux サーバーまたはクライアント上で、 `/usr/include/sys/socket.h` は AIX クライアント上にあります。 です。

If your NOF application needs to test against these values, you should use `#include` to include this system file in addition to the **ノード・ キュー・ ファイル** ヘッダー・ファイル。

アドレス・アドレス *.ip_addr.ipv4_addr* にレイプされました
このフィールドは、 家族 パラメーターが **アフリネット**に設定されている場合にのみ使用されます。 クライアントが CS Linux に接続するときに使用する IPv4 (ドット 10 進) アドレス。

addr.ip_addr.ipv6_addr.ipv6_addr.ipv6_addr
このフィールドは、 家族 パラメーターが **ネット受信 6** に設定されている場合にのみ使用されます。 クライアントが CS Linux に接続するために使用する IPv6 (コロン 16 進数) アドレス。

***client_adj_port* レイプ**
クライアントが CS Linux に接続するときに使用する IP ポート番号。

登録の失敗

REGISTRATION_FAILURE は、ネットワーク・ノード・サーバーにリソースを登録する試みが失敗したことを示します。

VCB structure

```
typedef struct registration_failure
{
   AP_UINT16        opcode;              /* verb operation code             */
   unsigned char    reserv2;             /* reserved                        */
   unsigned char    format;              /* reserved                        */
   AP_UINT16        primary_rc;          /* primary return code             */
   AP_UINT32        secondary_rc;        /* secondary return code           */
   unsigned char    data_lost;           /* previous indication lost        */
   unsigned char    resource_name[17];   /* network qualified resource name */
   AP_UINT16        resource_type;       /* resource type                   */
```

```
   unsigned char    description[32];      /* resource description           */
   unsigned char    reserv1[16];          /* reserved                       */
   unsigned char    reserv2b[2];          /* reserved                       */
   AP_UINT32        sense_data;           /* sense data                     */
   unsigned char    reserva[20];          /* reserved                       */
} REGISTRATION_FAILURE;
```

Parameters

opcode
AP_REGISTRATION_FAILURE

primary_rc
AP_OK

data_lost
Specifies whether any previous registration failure indications have been lost. If CS Linux detects a condition that prevents it from sending an indication (for example an internal resource shortage), it indicates this by setting the *data_lost* parameter on the next indication after the condition has cleared. Possible values are:

AP_YES
One or more previous registration failure indications were lost. Later fields in this VCB may be set to zeros.

AP_NO
No previous registration failure indications were lost.

resource_name
Name of resource that failed to register. The name is a 17-byte EBCDIC string, right-padded with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

resource_type
Resource type of resource that failed to register. One of the following.

AP_NNCP_RESOURCE
Network node.

AP_ENCP_RESOURCE
End node.

AP_LU_RESOURCE
LU.

description
A null-terminated text string describing the resource, as specified in the definition of the resource.

sense_data
Sense data (specified in SNA Formats).

RTP_指標

この指標は、次のいずれかが発生した場合に生成されます

- RTP 接続が接続または切断されています。
- アクティブなセッション・カウントが変更されます
- 接続は、パス・スイッチを実行します。

接続が切断されると、 RTP の最終統計が戻されます。 それ以外の場合は、 *rtp_stats* パラメーターは予約済みです。

VCB 構造体

```
typedef struct rtp_indication
{
   AP_UINT16        opcode;              /* Indication operation code      */
```

```
   unsigned char    reserv2;             /* reserved                         */
   unsigned char    format;              /* reserved                         */
   AP_UINT16        primary_rc;          /* primary return code              */
   AP_UINT32        secondary_rc;        /* secondary return code            */
   unsigned char    data_lost;           /* Previous indication lost?        */
   unsigned char    connection_state;    /* current state of the RTP         */
                                         /*  connection                      */
   unsigned char    rtp_name[8];         /* name of the RTP connection       */
   AP_UINT16        num_sess_active;     /* number of active sessions        */
   unsigned char    indication_cause;    /* reason for this indication       */
   unsigned char    connection_type;     /* usage of RTP connection          */
   unsigned char    reserv3[2];          /* reserved                         */
   RTP_STATISTICS   rtp_stats;           /* RTP statistics                   */
} RTP_INDICATION;
```

```
typedef struct rtp_statistics
{
   AP_UINT32        bytes_sent;           /* total number of bytes sent       */
   AP_UINT32        bytes_received;       /* total number of bytes received   */
   AP_UINT32        bytes_resent;         /* total number of bytes resent     */
   AP_UINT32        bytes_discarded;      /* total number of bytes discarded  */
   AP_UINT32        packets_sent;         /* total number of packets sent     */
   AP_UINT32        packets_received;     /* total number of packets received */
   AP_UINT32        packets_resent        /* total number of packets resent   */
   AP_UINT32        packets_discarded;    /* total number of packets discarded*/
   AP_UINT32        gaps_detected;        /* gaps detected                    */
   AP_UINT32        send_rate;            /* current send rate                */
   AP_UINT32        max_send_rate;        /* maximum send rate                */
   AP_UINT32        min_send_rate;        /* minimum send rate                */
   AP_UINT32        receive_rate;         /* current receive rate             */
   AP_UINT32        max_receive_rate;     /* maximum receive rate             */
   AP_UINT32        min_receive_rate;     /* minimum receive rate             */
   AP_UINT32        burst_size;           /* current burst size               */
   AP_UINT32        up_time;              /* total uptime of connection       */
   AP_UINT32        smooth_rtt;           /* smoothed round-trip time         */
   AP_UINT32        last_rtt;             /* last round-trip time             */
   AP_UINT32        short_req_timer;      /* SHORT_REQ timer duration         */
   AP_UINT32        short_req_timeouts;   /* number of SHORT_REQ timeouts     */
   AP_UINT32        liveness_timeouts;    /* number of liveness timeouts      */
   AP_UINT32        in_invalid_sna_frames; /* number of invalid SNA frames    */
                                           /*  received                        */
   AP_UINT32        in_sc_frames;         /* number of SC frames received     */
   AP_UINT32        out_sc_frames;        /* number of SC frames sent         */
   AP_INT32         delay_change_sum;     /* delay change sum                 */
   AP_UINT32        current_receiver_threshold;
                                          /* current ARB-R receiver threshold */
   AP_UINT32        minimum_receiver_threshold;
                                          /* minimum ARB-R receiver threshold */
   AP_UINT32        maximum_receiver_threshold;
                                          /* maximum ARB-R receiver threshold */
   AP_UINT32        sent_normals_count;   /* number of NORMALs sent           */
   AP_UINT32        sent_slowdowns_count; /* number of SLOWDOWNs sent         */
   AP_UINT32        rcvd_normals_count;   /* number of NORMALs received       */
   AP_UINT32        rcvd_slowdowns_count; /* number of SLOWDOWNs received     */
   AP_UINT32        dcs_reset_count_non_heal;
                                          /* number of non-healing resets     */
   AP_UINT16        dcs_reset_count_healing;
                                          /* number of self-healing resets    */
   unsigned char    arb_mode;             /* ARB mode (GREEN, YELLOW, RED)    */
   unsigned char    reserve[1];           /* reserved                         */
} RTP_STATISTICS;
```

パラメーター

オペコード
追加の P_TP_指標

primary_rc
アプオク

データ損失
前のディレクトリー指示が失われたかどうかを指定します。 CS Linux は、標識を送信できない状態 (例えば、内部リソース不足) を検出すると、条件がクリアされた後に次の指示に データ損失 パラメーターを設定することによって、これを示します。 可能な値は次のとおりです

類人猿
以前の 1 つ以上のディレクトリー指示が失われました。 この VCB の後のフィールドは、ゼロに設定することができます。

アプ・ノー
以前のディレクトリー指示は失われませんでした。

接続状態
RTP 接続の現在の状態。 可能な値は次のとおりです

接続中
接続セットアップが開始されましたが、まだ完了していません

接続済み
接続は完全にアクティブです。

非接続
接続はアクティブではなくなりました。

rtp_name
RTP 接続名。 この名前は、ローカルで表示可能な文字セットの 8 バイト・ストリングです。 8 バイトはすべて意味があります。

アクティブ状態の数
接続で現在アクティブになっているセッションの数。

指示原因
指示の原因。 可能な値は次のとおりです

非アクティブ化済み
接続がアクティブになりました。

非アクティブ化済み
接続が非アクティブになりました。

追加パス （ _R ）
接続がパス・スイッチを正常に完了しました。

AP_SESS_COUNT_変中
接続を使用しているアクティブ・セッションの数が変更されました。

AP_SETUP_FAILED
完全にアクティブになる 前に、接続が失敗しました。

接続タイプ
RTP 接続上のセッションのタイプを指定します。 可能な値は次のとおりです

アプリケーション・セッションの追加 CP_SESSION
RTP 接続は CP-CP セッションを伝送します。

追加 LU_LU_SESSION
RTP 接続は LU-LU セッションを伝送します。

AP_RTP_ROUTE_SETUP
RTP 接続は経路セットアップに使用されます。

以下のパラメーターは、接続が非活動状態になった場合 (指示原因 パラメーターが **非アクティブ化済み** または AP_SETUP_FAILED に設定されている場合) にのみ提供されます。 それ以外の場合は、以下のパラメーターが予約されています。

rtp_stats.bytes_sent
ローカル・ノードがこの RTP 接続で送信したバイトの合計数。

***rtp_stats.bytes_*譲り受けたバイト数**
ローカル・ノードがこの RTP 接続で受信したバイトの合計数。

rtp_stats.bytes_resent
転送中に損失が発生したためにローカル・ノードがこの RTP 接続上で再送した合計バイト数。

***rtp_stats.bytes_*捨てる**
既に受信されたデータの重複として廃棄された RTP 接続の相手側によって送信された合計バイト数。

送信された *rtp_stats.packets_sent*
ローカル・ノードがこの RTP 接続で送信したパケットの合計数。

受信した *rtp_stats.packets_受け取られた数*
ローカル・ノードがこの RTP 接続で受信したパケットの合計数。

rtp_stats.packets_resent
転送中に損失が発生したために、ローカル・ノードがこの RTP 接続上で再送したパケットの総数。

rtp_stats.packets_破棄
既に受信されたデータの重複として廃棄された RTP 接続の相手側によって送信されたパケットの合計数。

rtp_stats.gaps_検出された数
ローカル・ノードによって検出されたギャップの合計数。 各ギャップは、 1 つ以上の損失フレームに対応します。

rtp_stats.send_rate
この RTP 接続での現在の送信速度 (キロビット / 秒) 。 これは、 ARB アルゴリズムによって計算された最大許容送信速度です。

rtp_stats.max_send_rate
この RTP 接続での最大送信速度。 1 秒当たりの K ビット数で測定されます。

rtp_stats.min_send_rate
この RTP 接続での最小送信速度。 1 秒当たりの K ビット単位で測定されます。

rtp_stats.receive_rate
この RTP 接続での現在の受信速度 (キロビット / 秒) 。 これは、最後の計測間隔で計算された実際の受信率です。

rtp_stats.max_receive_rate
この RTP 接続での最大受信速度 (K ビット / 秒) 。

rtp_stats.min_receive_rate
この RTP 接続での最小受信速度。 1 秒当たりの K ビット単位で測定されます。

rtp_stats.burst_size
RTP 接続での現行バースト・サイズ。バイト単位で測定されます。

rtp_stats.up_time
RTP 接続がアクティブになっている 合計秒数。

rtp_stats.円滑化 rtt
ローカル・ノードとパートナー RTP ノードの間の往復時間の平滑化。ミリ秒で測定されます。

rtp_stats.last_rtt
ローカル・ノードとパートナー RTP ノードとの間の最後の測定往復時間 (ミリ秒) 。

rtp_stats.short_req_timer
SHORT_REQ タイマーに使用される現在の期間 (ミリ秒) 。

rtp_stats.short_req_timeouts
SHORT_REQ タイムアウトの数。

rtp_stats.liveness_timeouts
この RTP 接続で活性タイマーの有効期限が切れた合計回数。 *rtp_connection_detail.liveness_timer* で指定された接続のために接続がアイドル状態になると、活性タイマーが満了します。

rtp_stats.in_invalid_sna_フレーム
この RTP 接続では正しくない、受信および廃棄された SNA フレームの合計数。

rtp_stats.in_sc_frames
この RTP 接続で受信されたセッション制御フレームの合計数。

rtp_stats.out_sc_frames
この RTP 接続で送信されたセッション制御フレームの合計数。

rtp_stats.delay_change_sum
この RTP 接続で ARB-R アルゴリズムによって現在保持されている遅延変更合計の値。

rtp_stats.current_receiver_threshold
この RTP 接続で ARB-R アルゴリズムによって現在保持されている受信側しきい値の値。

rtp_stats.minimum_receiver_threshold
この RTP 接続で ARB-R アルゴリズムによって現在保持されている最小受信側しきい値の値。

rtp_stats.maximum_receiver_threshold
この RTP 接続で ARB-R アルゴリズムによって現在保持されている最大受信側しきい値の値。

rtp_stats.宣告された正常数
この RTP 接続で ARB-R アルゴリズムによって送信された NORMAL フィードバック ARB-R セグメントの数。

rtp_stats.sent_スローダウン数
この RTP 接続で ARB-R アルゴリズムによって送信された SLOWDOWN1 および SLOWDOWN2 フィードバック ARB-R セグメントの数。

rtp_stats.rcvd_normals_count
この RTP 接続で ARB-R アルゴリズムによって受信された NORMAL フィードバック ARB-R セグメントの数。

rtp_stats.rcvd_slowdowns_count
この RTP 接続で ARB-R アルゴリズムによって受信された SLOWDOWN1 および SLOWDOWN2 フィードバック ARB-R セグメントの数。

rtp_stats.dcs_reset_count_non_heal
この RTP 接続での通常 ARB-R 処理の一部として行われた遅延変更合計リセット数。

rtp_stats.dcs_reset_count_healing "
この RTP 接続で ARB-R アルゴリズムを自己修復するために行われた遅延変更合計のリセット数。

rtp_stats.arb_mode
この RTP 接続での現在の ARB-R 状況モード。 可能な値は次のとおりです

0
グリーン

1
イエロー

2
レッド

SERVER_INDICATION

This indication is generated when the CS Linux software is started or stopped on another computer on the LAN or when a server's role as controller or backup server changes. A NOF application can use these indications to keep track of which servers are currently active or to determine when a new server has successfully taken over as controller.

Server indications are also generated (for CS Linux internal use) when the status of other CS Linux components on a server changes. If the application needs to use server indications as described above, it should check the *status* and *flags* parameters for changes; it can ignore any server indications where these parameters do not indicate a change.

The REGISTER_INDICATION_SINK verb used to register for server indications should be issued with a null target handle; it is not associated with any particular target.

VCB structure

```
typedef struct server_indication
{
    AP_UINT16        opcode;               /* verb operation code          */
    unsigned char    reserv2;              /* reserved                     */
    unsigned char    format;               /* reserved                     */
    AP_UINT16        primary_rc;           /* primary return code          */
    AP_UINT32        secondary_rc;         /* secondary return code        */
    unsigned char    data_lost;            /* previous indication lost     */
    AP_UINT32        status;               /* node status                  */
```

```
    AP_UINT32        flags;              /* is server controller or backup?  */
    unsigned char    server_name[128];       /* name of server               */
} SERVER_INDICATION;
```

Parameters

opcode
AP_SERVER_INDICATION

primary_rc
AP_OK

data_lost
Specifies whether any previous server indications have been lost. If CS Linux detects a condition that prevents it from sending an indication (for example an internal resource shortage), it indicates this by setting the *data_lost* parameter on the next indication after the condition has cleared. Possible values are:

AP_YES
One or more previous server indications were lost. Later fields in this VCB may be set to zeros.

AP_NO
No previous server indications were lost.

status
Specifies the status of the SNA software on the indicated server. Possible values are:

AP_ACTIVE
The SNA software has been started.

AP_NOT_ACTIVE
The SNA software has been stopped.

flags
Specifies whether the indicated server is the controller server or a backup server. The application should use a logical AND operation to check the appropriate values, as follows:

- If the expression "*flags* AND AP_CONTROLER_FLAG" is nonzero, the indicated server is the controller server.
- If the expression "*flags* AND AP_BACKUP_FLAG" is nonzero, the indicated server is a backup server.

server_name
Name of the server on which the SNA software has been started or stopped.

セッション表示

この指示は、セッションがアクティブ化または非アクティブ化されると生成され セッションが非活動化されると、 verb はセッションの使用状況に関する統計を戻します。

VCB structure

```
typedef struct session_indication
{
   AP_UINT16         opcode;             /* verb operation code             */
   unsigned char     reserv2;            /* reserved                        */
   unsigned char     format;             /* reserved                        */
   AP_UINT16         primary_rc;         /* primary return code             */
   AP_UINT32         secondary_rc;       /* secondary return code           */
   unsigned char     data_lost;          /* previous indication lost        */
   unsigned char     deactivated;        /* has session been deactivated?   */
   unsigned char     lu_name[8];         /* LU name                         */
   unsigned char     lu_alias[8];        /* LU alias                        */
   unsigned char     plu_alias[8];       /* partner LU alias                */
   unsigned char     fqplu_name[17];     /* fully qualified partner LU name */
   unsigned char     mode_name[8];       /* mode name                       */
   unsigned char     session_id[8];      /* session ID                      */
   FQPCID            fqpcid;             /* fully qualified procedure       */
                                         /*  correlator ID                  */
   AP_UINT32         sense_data;         /* sense data                      */
```

```
   unsigned char     reserv1;            /* reserved                        */
   SESSION_STATS     sess_stats;         /* session statistics              */
   unsigned char     sscp_id[6];         /* SSCP ID                         */
   unsigned char     plu_slu_comp_lvl;   /* compression level PLU->SLU      */
   unsigned char     slu_plu_comp_lvl;   /* compression level SLU->PLU      */
   unsigned char     comp_in_series;     /* reserved                        */
   unsigned char     reserva[11];        /* reserved                        */
} SESSION_INDICATION;
```

```
typedef struct fqpcid
{
   unsigned char     pcid[8];            /* procedure correlator identifier */
   unsigned char     fqcp_name[17];      /* originator's network qualified  */
                                         /*  CP name                        */
   unsigned char     reserve3[3];        /* reserved                        */
} FQPCID;
```

```
typedef struct session_stats
{
  AP_UINT16        rcv_ru_size;          /* session receive RU size         */
  AP_UINT16        send_ru_size;         /* session send RU size            */
  AP_UINT16        max_send_btu_size;    /* maximum send BTU size           */
  AP_UINT16        max_rcv_btu_size;     /* maximum rcv BTU size            */
  AP_UINT16        max_send_pac_win;     /* maximum send pacing window size */
  AP_UINT16        cur_send_pac_win;     /* current send pacing window size */
  AP_UINT16        max_rcv_pac_win;      /* maximum receive pacing window   */
                                         /*  size                           */
  AP_UINT16        cur_rcv_pac_win;      /* current receive pacing window   */
                                         /*  size                           */
  AP_UINT32        send_data_frames;     /* number of data frames sent      */
  AP_UINT32        send_fmd_data_frames; /* num fmd data frames sent        */
  AP_UINT32        send_data_bytes;      /* number of data bytes sent       */
  AP_UINT32        rcv_data_frames;      /* number of data frames received  */
  AP_UINT32        rcv_fmd_data_frames;  /* num fmd data frames received    */
  AP_UINT32        rcv_data_bytes;       /* number of data bytes received   */
  unsigned char    sidh;                 /* session ID high byte            */
                                         /*  (from LFSID)                   */
  unsigned char    sidl;                 /* session ID low byte (from LFSID)*/
  unsigned char    odai;                 /* ODAI bit set                    */
  unsigned char    ls_name[8];           /* Link station name               */
  unsigned char    pacing_type;          /* Pacing type                     */
} SESSION_STATS;
```

Parameters

opcode
AP_SESSION_INDICATION

primary_rc
AP_OK

data_lost
Specifies whether any previous session indications have been lost. If CS Linux detects a condition that prevents it from sending an indication (for example an internal resource shortage), it indicates this by setting the *data_lost* parameter on the next indication after the condition has cleared. Possible values are:

AP_YES
One or more previous session indications were lost. Later fields in this VCB may be set to zeros.

AP_NO
No previous session indications were lost.

deactivated
Specifies whether the session has been deactivated or activated. Possible values are:

AP_YES
The session has been deactivated.

AP_NO
The session has been activated.

lu_name
LU name of the local LU, as defined to CS Linux. This is an 8-byte type-A EBCDIC string, padded on the right with spaces if the name is shorter than 8 bytes.

lu_alias
LU alias of the local LU, as defined to CS Linux. This is an 8-byte ASCII string, using any locally displayable characters, padded on the right with spaces if the name is shorter than 8 bytes.

plu_alias
LU alias of the partner LU. This is an 8-byte ASCII string, using any locally displayable characters, padded on the right with spaces if the name is shorter than 8 bytes.

fqplu_name
Fully qualified LU name for the partner LU, as defined to CS Linux. This name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of 1-8 A-string characters, an EBCDIC dot (period) character, and an LU name of 1-8 A-string characters.

mode_name
Name of the mode used by the LUs. This is an 8-byte alphanumeric type-A EBCDIC string (starting with a letter), padded on the right with EBCDIC spaces if the name is shorter than 8 bytes.

session_id
8-byte identifier of the session.

fqpcid.pcid
Procedure Correlator ID. This is an 8-byte hexadecimal string.

fqpcid.fqcp_name
Fully qualified CP name. The name is a 17-byte EBCDIC string, padded on the right with EBCDIC spaces. It consists of a network ID of up to 8 A-string characters, an EBCDIC dot (period) character, and a network name of up to 8 A-string characters.

The following parameters are used only if *deactivated* is set to AP_YES:

sense_data
The sense data sent or received on the UNBIND message that ended the session.

duplex_support
The conversation duplex support as negotiated on the BIND. Possible values are:

AP_HALF_DUPLEX
Only half-duplex conversations are supported.

AP_FULL_DUPLEX
Both half-duplex and full-duplex conversations are supported. Expedited data is also supported.

AP_UNKNOWN
Duplex support is not known because the session has deactivated.

sess_stats.rcv_ru_size
Maximum receive RU size.

sess_stats.send_ru_size
Maximum send RU size.

sess_stats.max_send_btu_size
Maximum BTU size that can be sent.

sess_stats.max_rcv_btu_size
Maximum BTU size that can be received.

sess_stats.max_send_pac_win
Maximum size of the send pacing window on this session.

sess_stats.cur_send_pac_win
Current size of the send pacing window on this session.

sess_stats.max_rcv_pac_win
Maximum size of the receive pacing window on this session.

sess_stats.cur_rcv_pac_win
Current size of the receive pacing window on this session.

sess_stats.send_data_frames
Number of normal flow data frames sent.

sess_stats.send_fmd_data_frames
Number of normal flow FMD data frames sent.

sess_stats.send_data_bytes
Number of normal flow data bytes sent.

sess_stats.rcv_data_frames
Number of normal flow data frames received.

sess_stats.rcv_fmd_data_frames
Number of normal flow FMD data frames received.

sess_stats.rcv_data_bytes
Number of normal flow data bytes received.

sess_stats.sidh
Session ID high byte.

sess_stats.sidl
Session ID low byte.

sess_stats.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.

sess_stats.ls_name
Link station name 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 session statistics with the link over which session traffic flows.

sess_stats.pacing_type
The type of receive pacing in use on this session.

sscp_id
For dependent LU sessions, the identifier of the SSCP as received in the ACTPU for the PU used by this LU. This parameter is 6 bytes and is used only by dependent LUs. This parameter is set to all zeros for independent LUs.

session_detail.plu_slu_comp_lvl
Specifies the compression level for data sent from the primary LU (PLU) to the secondary LU (SLU). Possible values are:

AP_NONE
Compression is not used.

AP_RLE_COMPRESSION
Run-length encoding (RLE) compression is used.

AP_LZ9_COMPRESSION
LZ9 compression is used.

AP_LZ10_COMPRESSION
LZ10 compression is used.

session_detail.slu_plu_comp_lvl
Specifies the compression level for data sent from the secondary LU (SLU) to the primary LU (PLU). Possible values are:

AP_NONE
Compression is not used.

AP_RLE_COMPRESSION
Run-length encoding (RLE) compression is used.

AP_LZ9_COMPRESSION
LZ9 compression is used.

AP_LZ10_COMPRESSION
LZ10 compression is used.

SNA_NET_INDICATION

This indication is generated when another NOF application or a CS Linux administration tool makes a change to the SNA network file `sna.net`. The target for this verb, identified by the *target_handle* parameter on the REGISTER_INDICATION_SINK verb that registers to receive this indication, must be the `sna.net` file.

VCB 構造体

この指示に関連付けられた特定の VCB 構造はありません。 SNA ネットワーク指示を登録するために、アプリケーションは、 REGISTER_INDICATION_SINK の 命令コード パラメーターとして値 SNA_NET_**指標の追加** を指定します。 SNA ネットワーク・ファイルへの変更が行われると、 CS Linux は、変更を行った NOF verb (ADD_BACKUP または DELETE_BACKUP) から VCB のコピーを送信することによって、アプリケーションのコールバック・ルーチンにこのことを報告します。

SNA ネットワーク・ファイルに発行された独自の NOF verb に対して、アプリケーションが SNA ネットワーク指示と非同期応答を区別できるようにするために、 CS Linux は、 VCB 内の *primary_rc* パラメーターを標識用に変更します。 値 **指示の表示** は、 SNA ネットワーク・ファイル指示に関連付けられた VCB を識別します。値 **アプオク**、またはその他の値は、アプリケーション独自の NOF verb の 1 つに対する非同期応答を示します。

TN_REDIRECTION_INDICATION

This indication is generated when a Telnet client starts or ends a session using TN Redirector. It is also generated when the SNA node providing TN Server function is stopped, to notify the application that it will need to re-register for TN Redirection indications; this is because registration for these indications is not maintained when the node stops and restarts.

VCB 構造体

```
typedef struct tn_redirection_indication
{
   AP_UINT16        opcode;                /* verb operation code            */
   unsigned char    reserv2;               /* reserved                       */
   unsigned char    format;                /* reserved                       */
   AP_UINT16        primary_rc;            /* primary return code            */
   AP_UINT32        secondary_rc;          /* secondary return code          */
   unsigned char    data_lost;             /* previous indication lost       */
   unsigned char    reason;                /* reason for indication          */
   SNA_IP_ADDR      client_ip_addr;        /* client IP address              */
   AP_UINT16        client_port;           /* client port number             */
   SNA_IP_ADDR      host_ip_addr;          /* host IP address                */
   AP_UINT16        host_port;             /* host port number               */
   unsigned char    client_number;         /* client number                  */
   unsigned char    listen_local_address[46];
                                           /* Local addr client connects to  */
   unsigned char    reserva[16];           /* reserved                       */
} TN_REDIRECTION_INDICATION;
```

```
typedef struct sna_ip_addr
{
   AP_UINT16        family;                  /* IPv4 or IPv6                 */
   union
   {
     unsigned char  ipv4_addr[4];
     unsigned char  ipv6_addr[16];
   } ip_addr;
} SNA_IP_ADDR;
```

パラメーター

オペコード
TN_REDIRECTION_指標

primary_rc
アプオク

データ損失
前の TN リダイレクト指示が失われたかどうかを指定します。 CS Linux は、標識を送信できない状態 (例えば、内部リソース不足) を検出すると、条件がクリアされた後に次の指示に データ損失 パラメーターを設定することによって、これを示します。 可能な値は次のとおりです

類人猿
1 つ以上の直前の TN リダイレクト指示が失われました。 この VCB の後のフィールドは、ゼロに設定することができます。

アプ・ノー
以前の TN リダイレクト指示は失われませんでした。

理由
この指示を送信する理由を指定します。 可能な値は次のとおりです

付加接続がアクティブ化されました
Telnet クライアントが TN リダイレクターを使用してセッションを開始しました。

付加接続が非アクティブ化されている
TN リダイレクター・セッションが終了しました。

TN_SERVER_終わっています
TN サーバー機能を提供するノードが停止しました。 このノードを使用しているアクティブな TN リダイレクター・セッションが存在する場合、アプリケーションは、 理由 が **付加接続が非アクティブ化されている**に設定されているセッションごとに、指示を受け取ります。

アプリケーションが、 TN リダイレクト指示の受信を続行する必要がある場合は、ノードの再始動時にこれらの指示を再登録する必要があります。

理由 が TN_SERVER_**終わっています**に設定されている場合、以下のフィールドは無効です。

クライアント *ip_addr.*ファミリー
Telnet クライアントが実行されるコンピューターに指定された TCP/IP アドレスのタイプ。 使用可能な値は以下のとおりです。 (これらは、 CS Linux によって定義される **追加 *** 値ではなく、標準 TCP/IP 値です。)

アフリネット
IPv4 アドレス。小数点付き 10 進数アドレス (193.1.11.100 など) として指定されます。

ネット受信 6
IPv6 アドレス。コロン付き 16 進アドレス (2001:0db8:0000:0000:0000:0000:0000:1428:57ab や 2001:db8::1428:57ab など) として指定します。

注 : 値 **アフリネット** と **ネット受信 6** は、システム・ヘッダー・ファイルから取得され、 CS Linux によって定義された標準の **追加 *** 値ではありません。 システム・ヘッダー・ファイルは /usr/include/linux/socket.h は、 Linux サーバーまたはクライアント上で、 /usr/include/sys/socket.h は AIX クライアント上にあります。 です。

If your NOF application needs to test against these values, you should use #include to include this system file in addition to the **ノード・キュー・ファイル** ヘッダー・ファイル。

client_ip_addr.ip_addr.ipv4_addr
このフィールドは、 クライアント *ip_addr.*ファミリー が **アフリネット**に設定されている場合のみ使用します。 Telnet クライアントが実行されているコンピューターの IPv4 (ドット 10 進) アドレス。

client_ip_addr.ip_addr.ipv6_addr
このフィールドは、 クライアント *ip_addr.*ファミリー が **ネット受信 6** に設定されている場合のみ使用します。 Telnet クライアントが実行されるコンピューターの IPv6 (コロン 16 進) アドレス。

クライアント・ポート
Telnet クライアントが TN リダイレクター・ノードへのアクセスに使用するサーバー TCP/IP ポートの番号。

ホスト *ip_addr*
クライアントが通信するホスト・コンピューターの TCP/IP アドレス。 これは、以下のいずれかになります。

- IPv4 小数点付き 10 進数アドレス (`193.1.11.100` など) 。
- IPv6 コロン 16 進アドレス (`2001:0db8:0000:0000:0000:0000:0000:0000:1428:57ab` や `2001:db8::1428:57ab` など) 。

ホスト・ポート
TN リダイレクター・ノードがホストへのアクセスに使用する TCP/IP ポートの番号。

クライアント番号
各クライアントに固有の番号。 これは、タイプ **付加接続がアクティブ化されました** の正常なリダイレクト指示を、タイプ **付加接続が非アクティブ化されている**のものと相関させるために使用できます。

listen_local_address
TN3270 クライアントが接続するローカル TN サーバー・コンピューター上のアドレス。

付録 A 戻りコードの値

この付録では、 NOF インターフェースの 可能なすべての戻りコードを番号順にリストします。 値はヘッダー・ファイルに定義されています 値 （ c.h ）.

この付録は、ご使用のアプリケーションが受け取った戻りコードの意味を確認するための参照として使用できます。

1 次戻りコード

以下の基本戻りコードが NOF アプリケーションで使用されます。

```
AP_OK                                0x0000
AP_PARAMETER_CHECK                   0x0100
AP_STATE_CHECK                       0x0200
AP_INDICATION                        0x0210
AP_TP_BUSY                           0x02F0
AP_ALLOCATION_ERROR                  0x0300
AP_ACTIVATION_FAIL_RETRY             0x0310
AP_COMM_SUBSYSTEM_ABENDED            0x03F0
AP_ACTIVATION_FAIL_NO_RETRY          0x0410
AP_COMM_SUBSYSTEM_NOT_LOADED         0x04F0
AP_DEALLOC_ABEND                     0x0500
AP_LU_SESS_LIMIT_EXCEEDED            0x0510
AP_DEALLOC_ABEND_PROG                0x0600
AP_FUNCTION_NOT_SUPPORTED            0x0610
AP_THREAD_BLOCKING                   0x06F0
AP_DEALLOC_ABEND_SVC                 0x0700
AP_DEALLOC_ABEND_TIMER               0x0800
AP_DATA_POSTING_BLOCKED              0x0810
AP_INVALID_VERB_SEGMENT              0x08F0
AP_DEALLOC_NORMAL                    0x0900
AP_PATH_SWITCH_NOT_ALLOWED           0x0910
AP_CP_CP_SESS_ACT_FAILURE            0x0A10
AP_PROG_ERROR_NO_TRUNC               0x0C00
AP_PROG_ERROR_TRUNC                  0x0D00
AP_PROG_ERROR_PURGING                0x0E00
AP_CONV_FAILURE_RETRY                0x0F00
AP_CONV_FAILURE_NO_RETRY             0x1000
AP_SVC_ERROR_NO_TRUNC                0x1100
AP_UNEXPECTED_DOS_ERROR              0x11F0
AP_SVC_ERROR_TRUNC                   0x1200
AP_SVC_ERROR_PURGING                 0x1300
AP_UNSUCCESSFUL                      0x1400
AP_STACK_TOO_SMALL                   0x15F0
AP_MIXED_API_USED                    0x16F0
AP_IN_PROGRESS                       0x17F0
AP_CNOS_PARTNER_LU_REJECT            0x1800
AP_COMPLETED                         0x18F0
AP_CONVERSATION_TYPE_MIXED           0x1900
AP_NODE_STOPPING                     0x1A00
AP_NODE_NOT_STARTED                  0x1B00
AP_CANCELLED                         0x2100
AP_BACKED_OUT                        0x2200
AP_DUPLEX_TYPE_MIXED                 0x2300
AP_LS_FAILURE                        0x2300
AP_OPERATION_INCOMPLETE              0x4000
AP_OPERATION_NOT_ACCEPTED            0x4100
AP_CONVERSATION_ENDED                0x4200
AP_ERROR_INDICATION                  0x4300
AP_EXPD_NOT_SUPPORTED_BY_LU          0x4400
AP_BUFFER_TOO_SMALL                  0x4500
AP_MEMORY_ALLOCATION_FAILURE         0x4600
AP_INVALID_VERB                      0xFFFF
```

Secondary return codes

The following secondary return codes are used in NOF applications.

```
AP_AS_SPECIFIED                         0x00000000
AP_ALLOCATION_ERROR_PENDING             0x00000300
AP_DEALLOC_ABEND_PROG_PENDING           0x00000600
AP_DEALLOC_ABEND_SVC_PENDING            0x00000700
AP_DEALLOC_ABEND_TIMER_PENDING          0x00000800
AP_UNKNOWN_ERROR_TYPE_PENDING           0x00001100
AP_BO_NO_RESYNC                         0x00002408
AP_TRANS_PGM_NOT_AVAIL_NO_RETRY         0x00004C08
AP_INVALID_SET_PROT                     0x00070000
AP_INVALID_DLUS_NAME                    0x00900000
AP_SEC_BAD_PASSWORD_EXPIRED             0x00FF0F08
AP_BAD_TP_ID                            0x01000000
AP_BO_RESYNC                            0x01002408
AP_INVALID_NEW_PROT                     0x01070000
AP_DLC_ACTIVE                           0x01100000
AP_NO_DEFAULT_DLUS_DEFINED              0x01900000
AP_BAD_TPSID                            0x01FF0000
AP_SEC_BAD_PASSWORD_INVALID             0x01FF0F08
AP_BAD_CONV_ID                          0x02000000
AP_SEND_ERROR_LOG_LL_WRONG              0x02010000
AP_INVALID_SET_UNPROT                   0x02070000
AP_INVALID_NUMBER_OF_NODE_ROWS          0x02080000
AP_DUPLICATE_CP_NAME                    0x02100000
AP_INVALID_PU_ID                        0x02900000
AP_NOT_OWNER                            0x02FF0000
AP_SEC_BAD_USERID_REVOKED               0x02FF0F08
AP_BAD_LU_ALIAS                         0x03000000
AP_BAD_DLOAD_ID                         0x03000001
AP_BAD_REMOTE_LU_ALIAS                  0x03000002
AP_SEND_ERROR_BAD_TYPE                  0x03010000
AP_INVALID_NEW_UNPROT                   0x03070000
AP_DUPLICATE_DEST_ADDR                  0x03100000
AP_PU_ALREADY_ACTIVATING                0x03900000
AP_INSUFFICIENT_PRIVILEGES              0x03FF0000
AP_SEC_BAD_USERID_INVALID               0x03FF0F08
AP_ALLOCATION_FAILURE_NO_RETRY          0x04000000
AP_SEND_ERROR_BAD_STATE                 0x04010000
AP_INVALID_SET_USER                     0x04070000
AP_NODE_ROW_WGT_LESS_THAN_LAST          0x04080000
AP_CANT_MODIFY_PORT_NAME                0x04100000
AP_PU_ALREADY_DEACTIVATING              0x04900000
AP_INVALID_CALLBACK                     0x04FF0000
AP_SEC_BAD_USERID_MISSING               0x04FF0F08
AP_ALLOCATION_FAILURE_RETRY             0x05000000
AP_BAD_ERROR_DIRECTION                  0x05010000
AP_INVALID_DATA_TYPE                    0x05070000
AP_TG_ROW_WGT_LESS_THAN_LAST            0x05080000
AP_DUPLICATE_PORT_NUMBER                0x05100000
AP_PU_ALREADY_ACTIVE                    0x05900000
AP_BAD_TP_TYPE                          0x05FF0000
AP_SEC_BAD_PASSWORD_MISSING             0x05FF0F08
AP_INVALID_STATS_TYPE                   0x06070000
AP_DUPLICATE_PORT_NAME                  0x06100000
AP_PU_NOT_ACTIVE                        0x06900000
AP_ALREADY_REGISTERED                   0x06FF0000
AP_SEC_BAD_GROUP_INVALID                0x06FF0F08
AP_AS_NEGOTIATED                        0x07000000
AP_INVALID_TABLE_TYPE                   0x07070000
AP_INVALID_DLC_NAME                     0x07100000
AP_DLUS_REJECTED                        0x07900000
AP_SEC_BAD_UID_REVOKED_IN_GRP           0x07FF0F08
AP_PORT_DEACTIVATED                     0x08070000
AP_INVALID_DLC_TYPE                     0x08100000
AP_DLUS_CAPS_MISMATCH                   0x08900000
AP_SEC_BAD_UID_NOT_DEFD_TO_GRP          0x08FF0F08
AP_ALLOCATE_NOT_PENDING                 0x09050000
AP_INVALID_SET_PASSWORD                 0x09070000
AP_INVALID_NUMBER_OF_TG_ROWS            0x09080000
AP_INVALID_LINK_ACTIVE_LIMIT            0x09100000
AP_PU_FAILED_ACTPU                      0x09900000
AP_SEC_BAD_UNAUTHRZD_AT_RLU             0x09FF0F08
AP_SNA_DEFD_COS_CANT_BE_CHANGE          0x0A080000
AP_SNA_DEFD_COS_CANT_BE_CHANGED         0x0A080000
AP_PU_NOT_RESET                         0x0A900000
AP_SEC_BAD_UNAUTHRZD_FROM_LLU           0x0AFF0F08
AP_INVALID_NUM_PORTS_SPECIFIED          0x0B100000
```

AP_PU_OWNS_LUS	0x0B900000
AP_SEC_BAD_UNAUTHRZD_TO_TP	0x0BFF0F08
AP_INVALID_PORT_NAME	0x0C100000
AP_INVALID_FILTER_OPTION	0x0C900000
AP_SEC_BAD_INSTALL_EXIT_FAILED	0x0CFF0F08
AP_INVALID_PORT_TYPE	0x0D100000
AP_INVALID_STOP_TYPE	0x0D900000
AP_SEC_BAD_PROCESSING_FAILURE	0x0DFF0F08
AP_UNRECOGNIZED_DEACT_TYPE	0x0E050000
AP_PORT_ACTIVE	0x0E100000
AP_PU_ALREADY_DEFINED	0x0E900000
AP_NO_PORTS_DEFINED_ON_DLC	0x0F100000
AP_DEPENDENT_LU_NOT_SUPPORTED	0x0F900000
AP_INVALID_DLC	0x10050000
AP_COS_NAME_NOT_DEFD	0x10080000
AP_DUPLICATE_PORT	0x10100000
AP_INVALID_DSPU_SERVICES	0x10900000
AP_BAD_CONV_TYPE	0x11000000
AP_SNA_DEFD_COS_CANT_BE_DELETE	0x11080000
AP_SNA_DEFD_COS_CANT_BE_DELETED	0x11080000
AP_STOP_PORT_PENDING	0x11100000
AP_DSPU_SERVICES_NOT_SUPPORTED	0x11900000
AP_BAD_SYNC_LEVEL	0x12000000
AP_LU_NAU_ADDR_ALREADY_DEFD	0x12020000
AP_INVALID_SESSION_ID	0x12050000
AP_LINK_DEACT_IN_PROGRESS	0x12100000
AP_INVALID_DSPU_NAME	0x12900000
AP_BAD_SECURITY	0x13000000
AP_INVALID_NN_SESSION_TYPE	0x13050000
AP_LINK_DEACTIVATED	0x13100000
AP_PARTNER_NOT_FOUND	0x13200000
AP_PARTNER_NOT_RESPONDING	0x13300000
AP_ERROR	0x13400000
AP_DSPU_ALREADY_DEFINED	0x13900000
AP_BAD_RETURN_CONTROL	0x14000000
AP_INVALID_MAX_NEGOT_SESS_LIM	0x14020000
AP_INVALID_SET_COLLECT_STATS	0x14050000
AP_LINK_ACT_BY_REMOTE	0x14100000
AP_INVALID_SOLICIT_SSCP_SESS	0x14900000
AP_INVALID_BACK_LEVEL_SUPPORT	0x15000000
AP_INVALID_MODE_NAME	0x15020000
AP_INVALID_SET_COLLECT_NAMES	0x15050000
AP_LINK_ACT_BY_LOCAL	0x15100000
AP_INVALID_TG_NUMBER	0x15500000
AP_MISSING_CP_NAME	0x15510000
AP_MISSING_CP_TYPE	0x15520000
AP_INVALID_CP_TYPE	0x15520000
AP_DUPLICATE_TG_NUMBER	0x15530000
AP_TG_NUMBER_IN_USE	0x15540000
AP_MISSING_TG_NUMBER	0x15550000
AP_PARALLEL_TGS_NOT_ALLOWED	0x15570000
AP_INVALID_BKUP_DLUS_NAME	0x15900000
AP_PIP_LEN_INCORRECT	0x16000000
AP_INVALID_RECV_PACING_WINDOW	0x16020000
AP_INVALID_SET_COLLECT_RSCVS	0x16050000
AP_SEC_REQUESTED_NOT_SUPPORTED	0x16900000
AP_NO_USE_OF_SNASVCMG	0x17000000
AP_INVALID_CNOS_SLIM	0x17020000
AP_LINK_NOT_DEFD	0x17100000
AP_INVALID_DUPLEX_SUPPORT	0x17900000
AP_UNKNOWN_PARTNER_MODE	0x18000000
AP_INVALID_TARGET_PACING_CNT	0x18020000
AP_PS_CREATION_FAILURE	0x18100000
AP_QUEUE_PROHIBITED	0x18900000
AP_INVALID_MAX_RU_SIZE_UPPER	0x19020000
AP_TP_ACTIVE	0x19100000
AP_INVALID_TEMPLATE_NAME	0x19900000
AP_INVALID_SNASVCMG_MODE_LIMIT	0x1A020000
AP_MODE_ACTIVE	0x1A100000
AP_CLASHING_NAU_RANGE	0x1A900000
AP_PLU_ACTIVE	0x1B100000
AP_INVALID_NAU_RANGE	0x1B900000
AP_INVALID_COS_SNASVCMG_MODE	0x1C020000
AP_INVALID_PLU_NAME	0x1C100000
AP_INVALID_NUM_DSLU_TEMPLATES	0x1C900000
AP_INVALID_DEFAULT_RU_SIZE	0x1D020000
AP_INVALID_SET_NEGOTIABLE	0x1D100000
AP_GLOBAL_TIMEOUT_NOT_DEFINED	0x1D900000
AP_INVALID_MIN_CONWINNERS	0x1E020000
AP_INVALID_MODE_NAME_SELECT	0x1E100000
AP_INVALID_RESOURCE_NAME	0x1E900000
AP_INVALID_RESPONSIBLE	0x1F100000

```
AP_INVALID_DLUS_RETRY_TIMEOUT           0x1F900000
AP_MODE_SESS_LIM_EXCEEDS_NEG            0x20020000
AP_INVALID_DRAIN_SOURCE                 0x20100000
AP_INVALID_DLUS_RETRY_LIMIT             0x20900000
AP_CPSVCMG_ALREADY_DEFD                 0x21020000
AP_INVALID_CN_NAME                      0x21080000
AP_INVALID_DRAIN_TARGET                 0x21100000
AP_TP_NAME_NOT_RECOGNIZED               0x21600810
AP_INVALID_MIN_CONLOSERS                0x21900000
AP_BAD_DUPLEX_TYPE                      0x22000000
AP_INVALID_BYPASS_SECURITY              0x22020000
AP_DEF_LINK_INVALID_SECURITY            0x22080000
AP_INVALID_FORCE                        0x22100000
AP_SYSTEM_TP_CANT_BE_CHANGED            0x22600810
AP_INVALID_MAX_RU_SIZE_LOW              0x22900000
AP_FDX_NOT_SUPPORTED_BY_LU              0x23000000
AP_TEST_INVALID_FOR_FDX                 0x23010000
AP_INVALID_IMPLICIT_PLU_FORBID          0x23020000
AP_INVALID_PROPAGATION_DELAY            0x23080000
AP_SYSTEM_TP_CANT_BE_DELETED            0x23600810
AP_INVALID_MAX_RECV_PACING_WIN          0x23900000
AP_SEND_EXPD_INVALID_LENGTH             0x24010000
AP_INVALID_SPECIFIC_SECURITY            0x24020000
AP_INVALID_EFFECTIVE_CAPACITY           0x24080000
AP_INVALID_CLEANUP_TYPE                 0x24100000
AP_INVALID_DYNAMIC_LOAD                 0x24600810
AP_RU_SIZE_LOW_UPPER_MISMATCH           0x24900000
AP_RCV_EXPD_INVALID_LENGTH              0x25010000
AP_INVALID_DELAYED_LOGON                0x25020000
AP_INVALID_COS_NAME                     0x25100000
AP_INVALID_ENABLED                      0x25600810
AP_LU_ALREADY_ACTIVATING                0x25900000
AP_EXPD_BAD_RETURN_CONTROL              0x26010000
AP_INVALID_CNOS_PERMITTED               0x26020000
AP_PW_SUB_NOT_SUPP_ON_SESS              0x26050000
AP_INVALID_SESSION_LIMIT                0x26100000
AP_INVALID_PIP_ALLOWED                  0x26600810
AP_LU_DEACTIVATING                      0x26900000
AP_EXPD_DATA_BAD_CONV_STATE             0x27010000
AP_INVALID_DRAIN                        0x27100000
AP_LU_ALREADY_ACTIVE                    0x27900000
AP_INVALID_PRLL_SESS_SUPP               0x28100000
AP_INVALID_MIN_CONTENTION_SUM           0x28900000
AP_INVALID_LU_NAME                      0x29100000
AP_COMPRESSION_NOT_SUPPORTED            0x29900000
AP_MODE_NOT_RESET                       0x2A100000
AP_INVALID_MAX_COMPRESS_LVL             0x2A900000
AP_MODE_RESET                           0x2B100000
AP_INVALID_COMPRESSION                  0x2B900000
AP_CNOS_REJECT                          0x2C100000
AP_INVALID_EXCEPTION_INDEX              0x2C900000
AP_INVALID_OP_CODE                      0x2D100000
AP_INVALID_MAX_LS_EXCEPTION             0x2D900000
AP_INVALID_DISABLE                      0x2E900000
AP_INVALID_MODIFY_TEMPLATE              0x2F900000
AP_INVALID_ALLOW_TIMEOUT                0x30900000
AP_CONFIRM_ON_SYNC_LEVEL_NONE           0x31000000
AP_PIP_NOT_ALLOWED                      0x31600810
AP_TRANS_PGM_NOT_AVAIL_RETRY            0x31604B08
AP_POST_ON_RECEIPT_BAD_FILL             0x31900000
AP_CONFIRM_BAD_STATE                    0x32000000
AP_UNKNOWN_USER                         0x32100000
AP_POST_ON_RECEIPT_BAD_STATE            0x32900000
AP_CONFIRM_NOT_LL_BDY                   0x33000000
AP_NO_PROFILES                          0x33100000
AP_INVALID_HPR_SUPPORT                  0x33900000
AP_CONFIRM_INVALID_FOR_FDX              0x34000000
AP_CONVERSATION_TYPE_MISMATCH           0x34600810
AP_INVALID_LU_MODEL                     0x34900000
AP_INVALID_MODEL_NAME                   0x35900000
AP_TOO_MANY_PROFILES                    0x36100000
AP_INVALID_CRYPTOGRAPHY                 0x36900000
AP_INVALID_UPDATE_TYPE                  0x37100000
AP_INVALID_CLU_CRYPTOGRAPHY             0x37900000
AP_DIR_ENTRY_PARENT                     0x38100000
AP_INVALID_RESOURCE_TYPES               0x38900000
AP_NODE_ALREADY_STARTED                 0x39100000
AP_CHECKSUM_FAILED                      0x39900000
AP_NODE_FAILED_TO_START                 0x3A100000
AP_DATA_CORRUPT                         0x3A900000
AP_LU_ALREADY_DEFINED                   0x3B100000
AP_INVALID_RETRY_FLAGS                  0x3B900000
```

AP_IMPLICIT_LU_DEFINED	0x3C100000
AP_DELAYED_VERB_PENDING	0x3C900000
AP_PORT_INACTIVE	0x3D100000
AP_DSLU_ACTIVE	0x3D900000
AP_ACTIVATION_LIMITS_REACHED	0x3E100000
AP_ACTIVATION_LIMITS_REACHED	0x3E100000
AP_INVALID_BRANCH_LINK_TYPE	0x3E900000
AP_PARALLEL_TGS_NOT_SUPPORTED	0x3F100000
AP_INVALID_BRNN_SUPPORT	0x3F900000
AP_DLC_INACTIVE	0x40100000
AP_BRNN_SUPPORT_MISSING	0x40900000
AP_CONFIRMED_BAD_STATE	0x41000000
AP_NO_LINKS_DEFINED	0x41100000
AP_SYNC_LEVEL_NOT_SUPPORTED	0x41600810
AP_INVALID_UPLINK	0x41900000
AP_CONFIRMED_INVALID_FOR_FDX	0x42000000
AP_STOP_DLC_PENDING	0x42100000
AP_INVALID_DOWNLINK	0x42900000
AP_INVALID_LS_ROLE	0x43100000
AP_INVALID_IMPLICIT_UPLINK	0x43900000
AP_INVALID_BTU_SIZE	0x44100000
AP_INVALID_ROCP_NAME	0x44900000
AP_LAST_LINK_ON_ACTIVE_PORT	0x45100000
AP_INVALID_REG_WITH_NN	0x45900000
AP_DYNAMIC_LOAD_ALREADY_REGD	0x46100000
AP_LS_PENDING_RETRY	0x46900000
AP_INVALID_LIST_OPTION	0x47100000
AP_INVALID_COS_TABLE_VERSION	0x47900000
AP_INVALID_RES_NAME	0x48100000
AP_CFRTP_REQUIRED_FOR_MLTG	0x48900000
AP_INVALID_RES_TYPE	0x49100000
AP_INVALID_MLTG_PAC_ALGORITHM	0x49900000
AP_INVALID_ADJ_NNCP_NAME	0x4A100000
AP_LIM_RESRCE_INVALID_FOR_MLTG	0x4A900000
AP_INVALID_NODE	0x4B100000
AP_AUTO_ACT_INVALID_FOR_MLTG	0x4B900000
AP_INVALID_ORIGIN_NODE	0x4C100000
AP_MLTG_LS_VISIBILITY_MISMATCH	0x4C900000
AP_INVALID_TG	0x4D100000
AP_SLTG_LINK_ACTIVE	0x4D900000
AP_INVALID_FQPCID	0x4E100000
AP_MLTG_LINK_PROPERTIES_DIFFER	0x4E900000
AP_INVALID_POOL_NAME	0x4F100000
AP_INVALID_ADJ_CP_NAME	0x4F900000
AP_BAD_TYPE	0x50020000
AP_INVALID_NAU_ADDRESS	0x50100000
AP_INVALID_ENABLE_POOL	0x50300000
AP_INVALID_SEND_TERM_SELF	0x50900000
AP_DEALLOC_BAD_TYPE	0x51000000
AP_LU_NAME_POOL_NAME_CLASH	0x51100000
AP_SECURITY_NOT_VALID	0x51600F08
AP_INVALID_TERM_METHOD	0x51900000
AP_DEALLOC_FLUSH_BAD_STATE	0x52000000
AP_INVALID_PRIORITY	0x52100000
AP_INVALID_DISABLE_BRANCH_AWRN	0x52900000
AP_DEALLOC_CONFIRM_BAD_STATE	0x53000000
AP_INVALID_DNST_LU_NAME	0x53100000
AP_INVALID_SHARING_PROHIBITED	0x53900000
AP_INVALID_HOST_LU_NAME	0x54100000
AP_INVALID_LINK_SPEC_FORMAT	0x54900000
AP_DEALLOC_NOT_LL_BDY	0x55000000
AP_PU_NOT_DEFINED	0x55100000
AP_INVALID_CN_TYPE	0x55900000
AP_INVALID_PU_NAME	0x56100000
AP_INVALID_PU_TYPE	0x56600000
AP_INCONSISTENT_BEST_EFFORT	0x56900000
AP_DEALLOC_LOG_LL_WRONG	0x57000000
AP_CNOS_MODE_NAME_REJECT	0x57010000
AP_INVALID_MAX_IFRM_RCVD	0x57100000
AP_INVALID_CN_TG	0x57900000
AP_INVALID_SYM_DEST_NAME	0x58100000
AP_SEC_BAD_PROTOCOL_VIOLATION	0x58600F08
AP_INVALID_LINK_SPEC_DATA	0x58900000
AP_INVALID_LENGTH	0x59100000
AP_DLC_UI_ONLY	0x59900000
AP_INVALID_ISR_THRESHOLDS	0x5A100000
AP_ADJ_CP_WRONG_TYPE	0x5A900000
AP_BAD_PARTNER_LU_ALIAS	0x5B010000
AP_INVALID_NUM_LUS	0x5B100000
AP_CP_CP_SESS_ALREADY_ACTIVE	0x5B900000
AP_EXCEEDS_MAX_ALLOWED	0x5C010000
AP_CANT_DELETE_ADJ_ENDNODE	0x5C100000

AP_NO_ACTIVE_CP_CP_LINK	0x5C900000
AP_LU_MODE_SESSION_LIMIT_ZERO	0x5D010000
AP_INVALID_RESOURCE_TYPE	0x5D100000
AP_PU_CONC_NOT_SUPPORTED	0x5E100000
AP_INVALID_IMPL_APPN_LINKS_LEN	0x5E900000
AP_CNOS_COMMAND_RACE_REJECT	0x5F010000
AP_DLUR_NOT_SUPPORTED	0x5F100000
AP_INVALID_LIMIT_ENABLE	0x5F900000
AP_INVALID_SVCMG_LIMITS	0x60010000
AP_INVALID_RTP_CONNECTION	0x60100000
AP_INVALID_LS_ATTRIBUTE	0x60900000
AP_FLUSH_NOT_SEND_STATE	0x61000000
AP_PATH_SWITCH_IN_PROGRESS	0x61100000
AP_HPR_NOT_SUPPORTED	0x62100000
AP_SOME_ENABLED	0x62900000
AP_RTP_NOT_SUPPORTED	0x63100000
AP_NONE_ENABLED	0x63900000
AP_COS_TABLE_FULL	0x64100000
AP_INCONSISTENT_IMPLICIT	0x64900000
AP_INVALID_DAYS_LEFT	0x65100000
AP_ANYNET_NOT_SUPPORTED	0x66100000
AP_INVALID_PERSIST_PIPE_SUPP	0x66900000
AP_INVALID_DISCOVERY_SUPPORT	0x67100000
AP_ACTIVATION_PROHIBITED	0x67900000
AP_SESSION_FAIL_ALREADY_REGD	0x68100000
AP_INVALID_NULL_ADDR_MEANING	0x68900000
AP_CANT_MODIFY_VISIBILITY	0x69100000
AP_INVALID_CPLU_SYNCPT_SUPPORT	0x69900000
AP_CANT_MODIFY_WHEN_ACTIVE	0x6A100000
AP_INVALID_CPLU_ATTRIBUTES	0x6A900000
AP_INVALID_BASE_NUMBER	0x6B100000
AP_INVALID_REG_LEN_SUPPORT	0x6B900000
AP_DEACT_CG_INVALID_CGID	0x6C020000
AP_INVALID_NAME_ATTRIBUTES	0x6C100000
AP_LUNAME_CGID_MISMATCH	0x6C900000
AP_NAU_ADDRESS_MISMATCH	0x6D100000
AP_INVALID_DDDLU_OFFLINE	0x6D900000
AP_POSTED_DATA	0x6E100000
AP_POSTED_NO_DATA	0x6F100000
AP_DEF_PLU_INVALID_FQ_NAME	0x74020000
AP_DLC_DEACTIVATING	0x86020000
AP_INVALID_WILDCARD_NAME	0x8C020000
AP_DUPLICATE	0x8D020000
AP_LU_NAME_WILDCARD_NAME_CLASH	0x8E020000
AP_INVALID_USERID	0x90020000
AP_INVALID_PASSWORD	0x91020000
AP_INVALID_PROFILE	0x93020000
AP_INVALID_TP_NAME	0xA0020000
AP_P_TO_R_INVALID_TYPE	0xA1000000
AP_INVALID_CONV_TYPE	0xA1020000
AP_P_TO_R_NOT_LL_BDY	0xA2000000
AP_P_TO_R_NOT_SEND_STATE	0xA3000000
AP_INVALID_SYNC_LEVEL	0xA3020000
AP_P_TO_R_INVALID_FOR_FDX	0xA5000000
AP_INVALID_LINK_NAME_SPECIFIED	0xB0020000
AP_RCV_AND_WAIT_BAD_STATE	0xB1000000
AP_INVALID_LU_ALIAS	0xB1020000
AP_RCV_AND_WAIT_NOT_LL_BDY	0xB2000000
AP_INVALID_NUM_LS_SPECIFIED	0xB2020000
AP_PLU_ALIAS_CANT_BE_CHANGED	0xB3020000
AP_PLU_ALIAS_ALREADY_USED	0xB4020000
AP_RCV_AND_WAIT_BAD_FILL	0xB5000000
AP_INVALID_AUTO_ACT_SUPP	0xB5020000
AP_CANT_DELETE_IMPLICIT_LU	0xB6020000
AP_FORCED	0xB7020000
AP_INVALID_LS_NAME	0xB7030000
AP_INVALID_LFSID_SPECIFIED	0xB7040000
AP_INVALID_FILTER_TYPE	0xB7050000
AP_INVALID_MESSAGE_TYPE	0xB7060000
AP_CANT_DELETE_CP_LU	0xB7070000
AP_ALL_RESOURCES_NOT_DEFINED	0xB7090000
AP_INVALID_LIST_TYPE	0xB70A0000
AP_RESOURCE_NAME_NOT_ALLOWED	0xB70B0000
AP_LU_ALIAS_CANT_BE_CHANGED	0xB8020000
AP_LU_ALIAS_ALREADY_USED	0xB9020000
AP_INVALID_LINK_ENABLE	0xBA020000
AP_INVALID_CLU_COMPRESSION	0xBB020000
AP_INVALID_DLUR_SUPPORT	0xBC020000
AP_ALREADY_STARTING	0xC0010000
AP_RCV_IMMD_BAD_STATE	0xC1000000
AP_INVALID_LINK_NAME	0xC1010000
AP_INVALID_USER_DEF_1	0xC3010000

```
AP_RCV_IMMD_BAD_FILL                0xC4000000
AP_INVALID_USER_DEF_2               0xC4010000
AP_INVALID_NODE_TYPE                0xC4020000
AP_INVALID_USER_DEF_3               0xC5010000
AP_INVALID_NAME_LEN                 0xC5020000
AP_INVALID_NETID_LEN                0xC6020000
AP_INVALID_NODE_TYPE_FOR_HPR        0xC8020000
AP_INVALID_MAX_DECOMPRESS_LVL       0xC9020000
AP_INVALID_CP_NAME                  0xCA010000
AP_INVALID_COMP_IN_SERIES           0xCA020000
AP_INVALID_LIMITED_RESOURCE         0xCE010000
AP_RCV_AND_POST_BAD_STATE           0xD1000000
AP_INVALID_BYTE_COST                0xD1010000
AP_RCV_AND_POST_NOT_LL_BDY          0xD2000000
AP_RCV_AND_POST_BAD_FILL            0xD5000000
AP_INVALID_TIME_COST                0xD6010000
AP_BAD_RETURN_STATUS_WITH_DATA      0xD7000000
AP_LOCAL_CP_NAME                    0xD7010000
AP_LS_ACTIVE                        0xDA010000
AP_INVALID_FQ_OWNING_CP_NAME        0xDB020000
AP_R_T_S_BAD_STATE                  0xE1000000
AP_R_T_S_INVALID_FOR_FDX            0xE2000000
AP_BAD_LL                           0xF1000000
AP_SEND_DATA_NOT_SEND_STATE         0xF2000000
AP_CP_OR_SNA_SVCMG_UNDELETABLE      0xF3010000
AP_SEND_DATA_INVALID_TYPE           0xF4000000
AP_DEL_MODE_DEFAULT_SPCD            0xF4010000
AP_SEND_DATA_CONFIRM_SYNC_NONE      0xF5000000
AP_MODE_NAME_NOT_DEFD               0xF5010000
AP_SEND_DATA_NOT_LL_BDY             0xF6000000
AP_MODE_UNDELETABLE                 0xF6010000
AP_SEND_TYPE_INVALID_FOR_FDX        0xF7000000
AP_INVALID_FQ_LU_NAME               0xFD010000
AP_INVALID_PARTNER_LU               0xFE010000
AP_INVALID_LOCAL_LU                 0xFF010000
```

付録 B 共通戻りコード

この付録では、すべての NOF verb に共通な 1 次戻りコードと 2 次戻りコードについて説明します。

特定の verb または動詞のグループに固有の戻りコードについては、37 ページの『第 3 章 NOF API verbs』の個々の動詞の説明で説明されています。

Communications subsystem not active

If the verb does not execute because a required component is not active, CS Linux returns the following parameters:

primary_rc
AP_COMM_SUBSYSTEM_ABENDED

secondary_rc
One of the following:

AP_LOCAL_ABENDED
The CS Linux software has stopped.

AP_TARGET_ABENDED
The target node has stopped or the communication path to it has failed.

primary_rc

AP_COMM_SUBSYSTEM_NOT_LOADED
The CS Linux software is not active.

secondary_rc
Not used.

primary_rc

AP_NODE_NOT_STARTED
The target node has not been started.

secondary_rc
Not used.

primary_rc

AP_NODE_STOPPING
The target node is in the process of stopping (as a result of a TERM_NODE verb).

secondary_rc
Not used.

表示

この戻りコードは、エラーを意味しません。

アプリケーションが、 REGISTER_INDICATION_SINK を使用して構成指示または SNA ネットワーク・ファイルの指示を受け取るように登録している場合、 CS Linux は、別の NOF API アプリケーションまたは CS Linux コンポーネントがターゲット・ファイルまたはターゲット・ノードの構成を変更するたびに、標識を送信します。 この指示の形式は、構成を変更した NOF verb の場合は、戻された VCB と同じです。 CS Linux は、この 1 次戻りコードを設定して、戻される VCB が、アプリケーションによって発行された verb への応答ではなく、構成標識または SNA ネットワーク・ファイルの指示であることを示します。これにより、アプリケーションは、他のアプリケーションによって発行された verb からの verb の戻りと指示を区別できるようになります。

primary_rc
指示の表示

secondary_rc
可能な値は次のとおりです

データの追加が失われています
CS Linux は、この指示の完全な VCB を戻すために十分なストレージを割り振ることができませんでした。戻された情報は不完全です。 アプリケーションは適切な QUERY_* verb を発行して、変更されたコンポーネントについての詳細情報を取得する必要があります。

(ゼロ)
この指示のための完全な VCB が戻されています。

Invalid function

If the verb does not execute because the node does not recognize it as a valid verb, CS Linux returns the following parameters:

primary_rc

AP_INVALID_VERB
The *opcode* parameter was not set to the operation code of any NOF verb, or the verb identified by this parameter cannot be used because this version of CS Linux does not support it.

secondary_rc
Not used.

primary_rc

AP_FUNCTION_NOT_SUPPORTED
The NOF verb identified by the specified *opcode* parameter cannot be used because the target node's configuration does not support it.

secondary_rc
Not used.

verb セグメントが無効です

VCB がデータ・セグメント内に含まれていなかったために verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc

ファイルの追加バー・ セグメント
データ・セグメントの終わりを超えて拡張された verb 制御ブロック。 verb は実行されませんでした。

2 次戻りコードは戻されません。

Parameter check

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
One of the following:

AP_INVALID_FORMAT
The reserved parameter *format* was not set to zero.

AP_INVALID_TARGET_HANDLE
The supplied target handle is not valid.

AP_INVALID_TARGET
The verb cannot be issued to the specified target. For example, QUERY_PARTNER_LU, which returns information about an LU's current usage, can be issued only to a running node; it is not valid when issued to a file.

AP_INVALID_TARGET_MODE
The verb cannot be issued in the current mode. For example, only QUERY_* verbs can be issued in read-only mode; DEFINE_*, DELETE_*, START_*, and STOP_* verbs are not valid in this mode.

AP_NOT_SERVER
This return code occurs only when you are running the NOF application program on a client. The verb that you issued is not valid on a client; it can be issued only on a server.

AP_SYNC_NOT_ENABLED
The application issued this verb within a callback routine, using the synchronous NOF entry point. Any verb issued from a callback routine must use the asynchronous entry point.

状態チェック

状態チェックのために verb が実行されない場合、 CS Linux は以下のパラメーターを戻します。

primary_rc
状態検査の追加

secondary_rc
以下のいずれか。

CANT_CANT_MODIFY_可視性
CS Linux によって内部的に使用されるように予約されている名前を持つリソースを定義しようとしました。 別の名前を選択してください。

ファイル・ロックを実行できませんでした
アプリケーションは、 SET_PROCESSING_MODE を発行してコミット・モードに変更しましたが、 CS Linux は、構成ファイルのロックを取得できませんでした。 これは、別の NOF API アプリケーションまたは CS Linux コンポーネントが既にファイルにアクセスしているためです。

ファイル・ファイルのロックが失敗しました
アプリケーションは、 SET_PROCESSING_MODE を発行して、コミット・モードから他のモードのいずれかに変更しましたが、 CS Linux は構成ファイルのロックを解除できませんでした。 このエラーが発生したときにファイルを解放するために、 CS Linux はアプリケーションのハンドルをファイルにクローズします。 アプリケーションは、新しいファイル・ハンドルを取得するために、再度 OPEN_FILE を発行する必要があります。これにより、このファイルへの verb の発行を試行します。

使用不可ファイルがあります
ターゲット・ファイルへの接続が失われました。

AP_NOT_コントローラー
ターゲット・ファイルは、コントローラー・サーバーではないサーバー上のドメイン構成ファイル、または **スネネット** ファイルのコピーです。 これらのファイルを変更する verb は、コントローラー・サーバーのファイルのコピーに対して発行される必要があります。

AP_SYNC_PENDING
この verb は、同期 NOF API エントリー・ポイントを使用して発行されましたが、別の同期 verb が進行中でした。 進行中の同期 verb は、いつでも 1 つしかありません。

System error

If the verb does not execute because of an operating system error, CS Linux returns the following parameters:

primary_rc

AP_UNEXPECTED_SYSTEM_ERROR
An operating system call failed during processing of the verb.

secondary_rc
The secondary return code in this case is the return code from the operating system call.

For the meaning of the operating system return code, see the file `/usr/include/errno.h` on the computer where the error occurred. Typically, the return code will indicate a condition such as memory shortage.

For the meaning of the operating system return code, refer to your operating system documentation.

If the problem persists, consult your System Administrator.

If the verb was issued to change the target configuration (such as DEFINE_* or DELETE_*), or to perform an action (such as START_*), the application should issue the appropriate QUERY_* verb to determine whether the change or action succeeded. In particular, if this error occurs while processing a DEFINE_* or DELETE_* verb containing multiple data structures, the change can be incomplete.

付録 C イブン へのコメントの送信方法

私たちの出版物に対するあなたの入力に感謝 情報の明確性、正確性、完全性についてのコメントや、お客様の他のフィードバックの提供については、無料でご意見ください。

以下の方法のいずれかを使用して、コメントを送信します。

1. Knowledge Center の下部にあるフィードバック・リンクを使用してください。
2. 以下のフィードバック・テンプレートを使用し、 "mhvrcfs@us.ibm.com" に E メールを送信してください。
3. コメントを以下のアドレスにメールします。

   株式会社 IBM
   注意: MHVRCFS リーダーのコメント
   707 号館 (日本)
   南道 2455 号線
   ポキプシー、 NY 12601–5400
   アス

E メールフィードバックテンプレート

以下のテンプレートを E メールにカット・アンド・ペーストしてください。 次に、必要な情報を入力します。

- 自分の名前:
- 会社、大学、または機関:
- コメントの対象となるトピックまたは Web ページの URL :
- コメントのテキスト

お客様のコメントについてお話しする場合は、電話番号を含め、最上の時間をお待ちください。

IBM にコメントを送信する場合、お客様に対していかなる義務も負うことなく、適切な方法でコメントを使用または配布するための非独占的な権利を IBM に付与します。

IBM またはその他の組織は、お客様が提供する個人情報を使用して、送信した問題についてのみお客様に連絡します。

技術的な問題がある場合

読者のコメントを送信するためにリストされているフィードバック・メソッドを使用しないでください。 代わりに、以下のいずれかのアクションを行ってください。

- 弊社技術員に連絡してください
- IBM テクニカル・サポートの連絡
- https://www.ibm.com/support/home/の IBM サポート・ポータルにアクセスしてください。

付録 D 特記事項

この情報は米国で提供される製品およびサービスについて作成されたものです。 本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。 お客様の地域で現在使用可能な製品およびサービスについては、 IBM 担当員にお尋ねください。 IBM 製品、プログラム、またはサービスに対するすべての参照は、その IBM 製品、プログラム、またはサービスのみが使用されることを意味するものではありません。 IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。 ただし、 IBM 以外の製品、プログラム、またはサービスの操作を評価および検証するのは、ユーザーの責任です。

IBM は、本書に記載されている内容について特許権 (特許出願中のものを含む) を この文書の提供により、これらの特許に対するライセンスは提供されません。 次の方法で、ライセンス照会を書面で送信することができます。

- IBM ライセンス・ディレクター
- 株式会社 IBM
- ノースキャッスル
- 法務・ NY 10504 ~ 1785
- 米国 .

2 バイト (DBCS) 情報に関するライセンス照会については、国内の IBM 知的財産省に連絡するか、書面でお問い合わせをお送りしてください。

- IBM ワールドトレードアジア 株式会社
- ライセンス交付
- 2-31 - 六本木三丁目、港区
- 日本東京 106

以下の規定は、英国その他の国に適用されないものとし、これらの規定が地方法に反する場合には適用しない。 内部ビジネス・マシンは、いかなる KIND、またはその他のいかなる保証 "現在のように" も負わないものとします。また、第三者の保証、商品性の保証、特定目的適合性の保証、または商品性の保証についての保証責任を負わないものとします。 いくつかの州では、保証責任の制限を許可しない場合があります。そのため、このステートメントはお客様に適用されない場合があります。

この情報には、技術的に不適切な記述や誤植が含まれている可能性があります。 本書には定期的に変更が加えられています。これらの変更内容は、本書の新しい版に組み込まれます。 IBM は予告なしに、随時、本資料に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書に記載されている IBM 以外の Web サイトへの参照は、便宜のために提供されており、それらの Web サイトの推奨事項としてはいかなる方法も提供していません。 これらの Web サイトの資料は、この IBM 製品の資料の一部ではありません。また、これらの Web サイトを使用することも、お客様自身のリスクに

IBM は、お客様が提供するいかなる情報も、お客様に対する義務を負うことのないものとして、自ら適切と信ずる方法で、使用または配布

本プログラムのライセンスにより、 (i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を提供することを希望する場合は、以下に連絡してください。

- 株式会社 IBM
- おや ボックス 12195
- コーンウォリス通り 3039 号
- リサーチ・トライアングル ・パーク (NC 27709-2195)
- 米国 .

かかる情報は、適切な使用条件 (場合によっては、手数料の支払いを含む) に従って使用できます。

本書で説明されているライセンス・プログラム、およびそれに使用可能なすべてのライセンス資料は、IBM お客様契約、 IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、 IBM によって提供されます。

ここに含まれるいかなるパフォーマンス・データも、管理環境下で決定されました そのため、他の稼働環境で得られた結果は、大幅に異なる場合があります。 一部の測定は、開発レベルのシステムで行われた可能性がありますが、これらの測定が一般的に使用可能なシステムで同じものになるという保証はありません。 さらに、一部の測定値が外挿によって推定されている場合があります 実際の結果は、異なる場合 この文書のユーザーは、それぞれの特定の環境に適したデータを検証する必要があります。

IBM 以外の製品に関する情報は、それらの製品の供給者、公開された発表、またはその他の公に利用可能なソースから入手したものです。 IBM はこれらの製品をテストしていないため、 IBM 以外の製品に関連するパフォーマンス、互換性、またはその他の請求の正確性を確認できません。 IBM 以外の製品の機能に関する質問は、それらの製品の供給者にお願いします。

この情報には、日常業務で使用されるデータおよびレポートの例が含まれています これらの例をできるだけ完全に説明するために、例には、個人、企業、ブランド、および製品の名前が含まれています。 これらの名前はすべて架空のものであり、実際のビジネス・エンタープライズによって 使用される名前と住所との類似性は、まったく偶発的なものです。

COPYRIGHT LICENSE: この情報には、さまざまなオペレーティング・プラットフォーム 上のプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で含まれています。 お客様は、サンプル・プログラムが作成されているオペレーティング・プラットフォームのアプリケーション ・プログラミング・インターフェースに 準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、 IBM に対価を支払うことなくこれらのサンプル・プログラムをコピー、変更、および配布することができます。 これらの例は、すべての条件下で完全にテストされているしたがって、 IBM は、これらのプログラムの信頼性、保守容易性、または機能を保証または実装することはできません。 IBM のアプリケーション・プログラミング・インターフェースに 準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、 IBM に対価を支払うことなくこれらのサンプル・プログラムをコピー、変更、および配布することができます。

これらのサンプル・プログラムまたは二次的著作物のそれぞれのコピーまたは部分には、次のように著作権表示を含める必要があります。 © (貴社の会社名) (年)。 このコードの一部は、 IBM Corp. から派生したものです。 サンプル・プログラム。 © 著作権 IBM Corp. の 2000 年、 2005 年、 2006 年、 2007 年、 2008 年、 2021 年。 すべての権限が予約済み

商標

以下は、 IBM Corporation の米国およびその他の国における商標または登録商標です。

- アクフ/フタム
- 拡張対等通信ネットワーキング (APPN)
- エイクス
- アプリケーション・システム /400
- アタン
- AS/400 システム
- シックス
- データベース 2
- ドブ 2
- エンタープライズ・システム /3090
- エンタープライズ・システム /4381
- エンタープライズ・システム /9000
- エエス /30 90
- エスエス 9000
- eServer
- イブン
- IBMLink
- イムス
- モフ
- MVS / ESA
- オペレーティング・システム /2
- オペレーティング・システム /400
- オウ号 2
- オー /400
- パワー PC
- PowerPC アーキテクチャー
- システム /390
- システム /390
- システム p5
- System z
- システム z9
- ヴァース/エサ
- ヴタム
- WebSphere

・その他、記載されている会社名、製品名は、各社の登録商標

Java およびすべての Java 関連の商標およびロゴは、 Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

UNIX は、 The Open Group が独占的にライセンスしている米国およびその他の国における登録商標です。

Intel 、 EM64T は、 Intel Corporation の商標または登録商標です。

・ AMD64 は、 Advanced Micro Devices, Inc. の商標です。

Linux は、 Linus Torvalds 氏の商標です。

RedHat は、 Red Hat, Inc. の商標または登録商標です。

SuSE Linux は、 Novell の商標です。

Ubuntu は、 Canonical Limited の商標である。

Microsoft 、 Windows 、 Windows Server 、 Windows Server は、米国 Microsoft Corporation の米国およびその他の国における商標または登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標または登録商標です。

Bibliography

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

- CS Linux, Version 7.1
- 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 CS Linux library, brief descriptions are provided. For other books, only the titles and order numbers are shown here.

CS Linux version 7.1 publications

The CS Linux library comprises the following books. In addition, softcopy versions of these documents are provided on the CD-ROM. See *IBM Communications Server for Data Center Deployment on 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 Data Center Deployment on Linux Quick Beginnings* (GC31-6768 and GC31-6769)

  This book is a general introduction to CS Linux, including information about supported network characteristics, installation, configuration, and operation. There are two versions of this book:

  – GC31-6768 is for CS Linux on the i686, x86_64, and ppc64 platforms

  – GC31-6769 is for CS Linux for IBM Z.
- *IBM Communications Server for Data Center Deployment on Linux Administration Guide* (SC31-6771)

  This book provides an SNA and CS Linux overview and information about CS Linux configuration and operation.
- *IBM Communications Server for Data Center Deployment on Linux Administration Command Reference* (SC31-6770)

  This book provides information about SNA and CS Linux commands.
- *IBM Communications Server for Data Center Deployment on AIX or Linux CPI-C Programmer's Guide* (SC23-8591)

  This book provides information for experienced `C' or Java™programmers about writing SNA transaction programs using the CS Linux CPI Communications API.
- *IBM Communications Server for Data Center Deployment on AIX or Linux APPC Programmer's Guide* (SC23-8592)

  This book contains the information you need to write application programs using Advanced Program-to-Program Communication (APPC).
- *IBM Communications Server for Data Center Deployment on AIX or Linux LUA Programmer's Guide* (SC23-8590)

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

- *IBM Communications Server for Data Center Deployment on AIX or Linux CSV Programmer's Guide* (SC23-8589)

  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 Data Center Deployment on AIX or Linux MS Programmer's Guide* (SC23-8596)

  This book contains the information you need to write applications using the Management Services (MS) API.

- *IBM Communications Server for Data Center Deployment on Linux NOF Programmer's Guide* (SC31-6778)

  This book contains the information you need to write applications using the Node Operator Facility (NOF) API.

- *IBM Communications Server for Data Center Deployment on Linux Diagnostics Guide* (SC31-6779)

  This book provides information about SNA network problem resolution.

- *IBM Communications Server for Data Center Deployment on AIX or Linux APPC Application Suite User's Guide*(SC23-8595)

  This book provides information about APPC applications used with CS Linux.

- *IBM Communications Server for Data Center Deployment on Linux Glossary* (GC31-6780)

  This book provides a comprehensive list of terms and definitions used throughout the CS Linux library.

Systems Network Architecture (SNA) publications

The following books contain information about SNA networks:

- システム・ネットワーク体系*:* フォーマットおよびプロトコル参照マニュアル *- LU* タイプ *6.2* のためのアーキテクチャー・ロジック (SC30-3269)
- システム・ネットワーク体系*:* フォーマット (GA27-3136)
- システム・ネットワーク・アーキテクチャー*: SNA* 資料への手引き (GC30-3438)
- システム・ネットワーク体系*:* ネットワーク・プロダクト・フォーマット (LY43-0081)
- システム・ネットワーク・アーキテクチャー*:* テクニカル概要 (GC30-3073)
- システム・ネットワーク体系*: APPN* アーキテクチャー参照 (SC30-3422)
- システム・ネットワーク体系*:* 論理装置間のセッション (GC20-1868)
- システム・ネットワーク体系*: LU 6.2* 参照 - ピア・プロトコル (SC31-6808)
- システム・ネットワーク体系*: LU* タイプ *6.2* のためのトランザクション・プログラマーズ・リファレンス・マニュアル (GC30-3084)
- システム・ネットワーク体系*: 3270* データ・ストリーム・プログラマーズ・リファレンス (GA23-0059)
- ネットワーキング・ブループリント ・エグゼクティブの概要 (GC31-7057)
- システム・ネットワーク体系*:* 管理サービス参照 (SC30-3346)

Host configuration publications

The following books contain information about host configuration:

- *ES/9000*、 *ES/3090 IOCP* ユーザーズ・ガイド・ボリューム *A04* (GC38-0097)
- *3174* 確立コントローラー・インストール・ガイド (GG24-3061)
- *3270* 情報表示システム *3174* 確立コントローラー*:* 計画ガイド (GA27-3918)
- *OS/390* ハードウェア構成定義 *(HCD)* ユーザーズ・ガイド (SC28-1848)

z/OS Communications Server publications

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

- *z/OS V1R7* コミュニケーション・サーバー*: SNA* ネットワーク実装ガイド (SC31-8777)
- *z/OS V1R7* 通信サーバー*: SNA* 診断 (Vol 1: GC31-6850, Vol 2: GC31-6851)
- *z/OS V1R6* 通信サーバー*:* リソース定義リファレンス (SC31-8778)

TCP/IP publications

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

- *z/OS V1R7* コミュニケーション・サーバー*: IP* 構成ガイド (SC31-8775)
- *z/OS V1R7* 通信サーバー*: IP* 構成参照 (SC31-8776)
- *z/VM V5R1* の *TCP/IP* の計画およびカスタマイズ (SC24-6125)

X.25 の資料

以下の資料には、 X.25 ネットワーク・プロトコルに関する情報が含まれて

- *OS/2* バージョン *4 X.25* プログラミングの *Communications Server* (SC31-8150)

APPC の資料

以下の資料には、 Advanced Program-to-Program Communication (APPC)に関する情報が含まれています。

- *APPC* アプリケーション・スイート *V1* ユーザーズ・ガイド (SC31-6532)
- *APPC* アプリケーション・スイート *V1* の管理 (SC31-6533)
- *APPC* アプリケーション・スイート *V1* のプログラミング (SC31-6534)
- *APPC* アプリケーション・スイート *V1* オンライン・プロダクト・ライブラリー (SK2T-2680)
- *APPC* アプリケーション・スイート・ライセンス・プログラムの仕様 (GC31-6535)
- *z/OS V1R2.0 Communications Server: APPC* アプリケーション・スイート・ユーザーズ・ガイド (SC31-8809)

プログラミングの資料

以下の資料には、プログラミングに関する情報が記載

- 共通プログラミング・インターフェース 通信 *CPI-C* リファレンス (SC88-498)
- *OS/2* バージョン *4* アプリケーション・プログラミング・ガイドの *Communications Server* (SC31-8152)

Other IBM networking publications

The following books contain information about other topics related to CS Linux:

- *SDLC* の概念(GA27-3093)
- ローカル・エリア・ネットワークの概念および製品*: LAN* アーキテクチャー (SG24-4753)
- ローカル・エリア・ネットワークの概念および製品*: LAN* アダプター、ハブおよび *ATM* (SG24-4754)
- ローカル・エリア・ネットワークの概念および製品*:* ルーターおよびゲートウェイ (SG24-4755)
- ローカル・エリア・ネットワークの概念および製品*: LAN* オペレーティング・システムおよび管理 (SG24-4756)

- *IBM* ネットワーク制御プログラム・リソース定義ガイド (SC30-3349)

Index

Special Characters

Numerics

A

B

C

D

E

F

H

I

K

L

M

N

P

Q

R

S

T

U

V

SC31-6778-05